

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Shinichiro HAMADA, et al.

GAU:

SERIAL NO: NEW APPLICATION

EXAMINER:

FILED: HEREWITH

FOR: METHOD AND APPARATUS FOR EDITING WEB DOCUMENT FROM PLURALITY OF WEB SITE  
INFORMATION

#2  
11017 U.S. PTO  
10/015604  
12/17/01

REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS  
WASHINGTON, D.C. 20231

SIR:

- ☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of 35 U.S.C. §120.
- ☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of 35 U.S.C. §119(e).
- ☒ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of 35 U.S.C. §119, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY

APPLICATION NUMBER

MONTH/DAY/YEAR

Japan

2000-383625

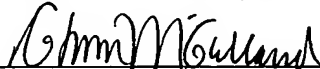
December 18, 2000

Certified copies of the corresponding Convention Application(s)

- ☒ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee
- ☐ were filed in prior application Serial No. filed
- ☐ were submitted to the International Bureau in PCT Application Number  
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.
- ☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and
- ☐ (B) Application Serial No.(s)
- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,  
MAIER & NEUSTADT, P.C.



Marvin J. Spivak

Registration No. 24,913

C. Irvin McClelland  
Registration Number 21,124



22850

日 本 国 特 許 庁  
JAPAN PATENT OFFICE

J1017 U.S. PTO  
10/015604  
12/17/01

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2000年12月18日

出 願 番 号

Application Number:

特願2000-383625

出 願 人

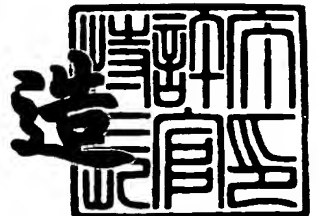
Applicant(s):

株式会社東芝

2001年 8月31日

特許庁長官  
Commissioner,  
Japan Patent Office

及川耕造



出証番号 出証特2001-3079866

【書類名】 特許願

【整理番号】 A000007604

【提出日】 平成12年12月18日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 7/00

【発明の名称】 文書合成方法および文書合成装置

【請求項の数】 8

【発明者】

【住所又は居所】 神奈川県川崎市幸区小向東芝町1番地 株式会社東芝研究開発センター内

【氏名】 浜田 伸一郎

【発明者】

【住所又は居所】 神奈川県川崎市幸区小向東芝町1番地 株式会社東芝研究開発センター内

【氏名】 關 俊文

【特許出願人】

【識別番号】 000003078

【氏名又は名称】 株式会社 東芝

【代理人】

【識別番号】 100058479

【弁理士】

【氏名又は名称】 鈴江 武彦

【電話番号】 03-3502-3181

【選任した代理人】

【識別番号】 100084618

【弁理士】

【氏名又は名称】 村松 貞男

【選任した代理人】

【識別番号】 100068814

【弁理士】

【氏名又は名称】 坪井 淳

【選任した代理人】

【識別番号】 100092196

【弁理士】

【氏名又は名称】 橋本 良郎

【選任した代理人】

【識別番号】 100091351

【弁理士】

【氏名又は名称】 河野 哲

【選任した代理人】

【識別番号】 100088683

【弁理士】

【氏名又は名称】 中村 誠

【選任した代理人】

【識別番号】 100070437

【弁理士】

【氏名又は名称】 河井 将次

【手数料の表示】

【予納台帳番号】 011567

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 文書合成方法および文書合成装置

【特許請求の範囲】

【請求項 1】 インターネットにおける WWW (World Wide web) 上のマークアップ言語で記述された複数の第 1 の文書の内容の一部を WWW 上のマークアップ言語で記述された第 2 の文書に合成するための文書合成方法であって、

少なくとも、前記第 1 の文書の該インターネット上の所在と、該第 1 の文書から抽出する部分文書の範囲と、前記第 2 の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第 2 の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第 2 の文書に従って、

前記第 1 の文書から前記部分文書を抽出して、その部分文書を前記第 2 の文書上の前記指定された合成位置に挿入するとともに、前記変換ルールを用いて前記第 2 の文書上の前記指定された範囲の文書構造を変換することで、前記第 2 の文書上に 1 または複数の前記部分文書を合成することを特徴とする文書合成方法。

【請求項 2】 前記第 2 の文書は、少なくとも、前記第 2 の文書上の前記部分文書の挿入位置とを指定するとともに、前記第 1 の文書の所在と、該第 1 の文書から抽出する部分文書の範囲とを記述するため第 1 のタグと、

前記変換ルールを用いて文書構造を変換すべき範囲を指定するとともに、前記変換ルールを記述したファイルの識別情報を記述するための第 2 のタグと、

を用いて記述されていることを特徴とする請求項 1 記載の文書合成方法。

【請求項 3】 前記第 2 の文書は、XML (Extensible Markup Language) で記述され、前記第 1 の文書が XML で記述されていないときは、まず、XML による記述型式に変換した後、前記第 1 の文書から前記部分文書を抽出して、その部分文書を前記第 2 の文書上の前記指定された挿入位置に挿入することを特徴とする請求項 1 記載の文書合成方法。

【請求項 4】 インターネットにおける WWW (World Wide web

e b) 上のマークアップ言語で記述された複数の第 1 の文書の内容の一部を WWW 上のマークアップ言語で記述された第 2 の文書に合成する文書合成装置であって、

少なくとも、前記第 1 の文書の該インターネット上の所在と、該第 1 の文書から抽出する部分文書の範囲と、前記第 2 の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第 2 の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第 2 の文書に従って、前記第 1 の文書から前記部分文書を抽出して、その部分文書を前記第 2 の文書上の前記指定された挿入位置に挿入する挿入手段と、

前記第 2 の文書に従って、該第 2 の文書上の前記指定された範囲の文書構造を、前記変換ルールを用いて所望の文書構造に変換する変換手段と、

を具備し、

前記第 2 の文書上に 1 または複数の前記部分文書を合成することを特徴とする文書合成装置。

【請求項 5】 前記第 2 の文書は、少なくとも、前記第 2 の文書上の前記部分文書の挿入位置とを指定するとともに、前記第 1 の文書の所在と、該第 1 の文書から抽出する部分文書の範囲とを記述するため第 1 のタグと、

前記変換ルールを用いて文書構造を変換すべき範囲を指定するとともに、前記変換ルールを記述したファイルの識別情報を記述するための第 2 のタグと、

を用いて記述されていることを特徴とする請求項 4 記載の文書合成装置。

【請求項 6】 前記第 2 の文書は、XML (Extensible Markup Language) で記述されていることを特徴とする請求項 4 記載の文書合成装置。

【請求項 7】 前記第 1 の文書が XML で記述されていないとき、該第 1 の文書を XML による記述型式に変換する第 2 の変換手段をさらに具備し、

前記挿入手段は、XML 文書の前記第 1 の文書から前記部分文書を抽出して、その部分文書を前記第 2 の文書上の前記指定された挿入位置に挿入することを特徴とする請求項 4 記載の文書合成装置。

【請求項 8】 インターネットにおける WWW (World Wide web) 上のマークアップ言語で記述された複数の第 1 の文書の内容の一部をマークアップ言語で記述された第 2 の文書に合成するための処理をコンピュータに実行させるためのプログラムであって、

少なくとも、前記第 1 の文書の該インターネット上の所在と、該第 1 の文書から抽出する部分文書の範囲と、前記第 2 の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第 2 の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第 2 の文書に従って、前記第 1 の文書から前記部分文書を抽出して、その部分文書を前記第 2 の文書上の前記指定された挿入位置に挿入するための処理と、

前記第 2 の文書に基づき、該第 2 の文書上の前記指定された範囲の文書構造を、前記変換ルールを用いて所望の文書構造に変換するための処理と、

をコンピュータに実行させるためのプログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、複数のウェブ文書を 1 つのウェブ文書上に合成するためのウェブ文書合成方法およびそれを用いたウェブ文書合成装置に関する。

【0002】

【従来の技術】

WWW (World Wide Web) は効果的なプレゼンテーションを低コストで構築・公開できる情報基盤として普及し、世界中のサイトで膨大な情報資源が公開されている。また WWW はサーバクライアントシステムのためのインフラの側面を持っている。特に電子商取引や最近では ASP (Application Service Providing) などへの応用が期待されており、本格的なコマースサイトが急増しつつある状況にある。電子商取引では、ウェブページは、商取引を処理する企業内 LAN のバックエンドシステムとユーザとを結ぶ操作パネルとしての役割を果たす。WWW はサイトを越えて世界中のコン

ピュータシステムをつなぐ唯一のインフラであるが、今後もウェブトップ指向への流れは続くことが予想される。

【0003】

WWWで交換される情報資源は増加の一途をたどり、ウェブシステムに要求される処理はより複雑で多様なものになるだろう。

【0004】

特に、企業はWWWを積極的に活用しており、企業データやニュース・商品カタログ情報など自社の持つ大量のデータをウェブページを通じて公開しているが、各ウェブページを一から作るにはあまりにも人手がかかりすぎるため、定型的なコンテンツを含むウェブページについては、データベースから静的あるいは動的に機械生成する技術を導入しており、サイト構築および運用を効率化している。このようなウェブサイトの構築・運用ツールは、多くのソフトウェアベンダーから提供されており、非常に充実している。しかしこれらの技術はいずれも閉じた単一ウェブサイトの構築や運用の効率化・高性能化に関するものである。

【0005】

単一ウェブサイトの構築・運用環境が整備された現在、次にWWWに求められるのはウェブサイト間連携である。すなわちサーバクライアントシステムから分散システムへの発展である。特に本格的な電子商取引の時代を迎えるにあたり、各コマースサイトの電子商取引システムの連携は必須となる。

【0006】

電子商取引システムの連携には、商品プロフィールなどのデータフォーマットや語彙の共通化、そして共通のビジネスモデル、それに従った共通のメッセージフォーマットやプロトコルなど多くの取り決めが必要である。これに対し、OASISやBizTalkなど業界団体が標準化を進めているが、企業間の利害の不一致や商習慣の違いなど多くの壁があるため、その成果が実を結ぶには、まだまだ時間を要することは間違いない。

【0007】

一方でその火急のニーズに対応するため、各ソフトウェアベンダーからは、上述のウェブサイト構築・運用ツールにウェブサイトの連携機構を追加したパッケ



ージが提供されている。

【0008】

しかし、データベースを中心に据えたアプリケーションロジック群を核とする従来のシステム構築手法は、単一ウェブサイトに対してはウェブページを単なるユーザインターフェースとして位置付けることで有効に機能したが、複数ウェブサイトにもたがるシステムに対してはそのままでは適用できない。なぜなら、この構築手法ではシステム連携を実現するためにアプリケーションロジックを接続する必要があるが、サイト間はファイアウォールによってさえぎられており、ほとんどの場合HTTP以外のメッセージが交換できないからである。

【0009】

従って、唯一のメッセージ交換のチャンネルであるHTTPをベースとしたシステム統合モデルが必要だが、パッケージの多くは従来のサイト構築技術にHTTPアクセス機能を追加しただけであり、HTTPおよびWWWの機能を生かしてきれていない状況にある。

【0010】

このようにサイト間のシステム連携は、それぞれのシステムが持つロジックを接続するために多くの取り決めが必要であり本質的に難しい課題である。

【0011】

そこで、ロジック接続ではなくコンテンツ交換を用いたウェブサイト間連携を課題として着目してみると、ウェブサイト間コンテンツ連携は、ウェブリソースの構造変換程度の調節ですむため、ウェブサイト間システム連携に比べて解決すべき課題は少ない。

【0012】

しかし、その一方で、コンテンツ連携がもたらす効果は十分に大きい。先に述べたようにWWWではすでに膨大なウェブリソースが公開されている。またウェブリソースはマルチメディアであり、あらゆるコンテンツメディアを包括することができる。このようなウェブリソースをサイト間で合意の下に互いに容易に再利用できる環境があれば、WWWは格段に合理的で経済的なものになり、WWWの応用に大きな進歩をもたらすだろう。

## 【0013】

例えば、本の売上情報やTV番組の視聴率情報など、ウェブサイトを構成する情報資源の一部をアウトソーシングするといった、分散管理型のウェブサイト構築スタイルが可能となり、大きなウェブパーツ市場が生まれる可能性もある。また、各ショッピングサイトが抱える商品カタログを1つのウェブページ上で比較表示するショッピングモールや、複数の調達システムやオークションシステムなどが抱える案件を統合したマーケットプレイスなどの仲介サービスを行うポータルサイトが最近次々と登場してきており非常に注目されている。これはウェブ情報が非常に氾濫してきている情勢においてウェブ情報を整理したり案内役を果たすサービスへ必然的なニーズが高まっているからであり、その要求に応える一つの形である。ウェブリソースを互いに再利用するための環境整備は、このようなポータルサイトの構築に大きな貢献をするだろう。その視点から、電子商取引システムなどウェブサイト間システム連携への足がかりとなる着実な技術移行という位置付けとも言える。

## 【0014】

さて、ウェブページ検索サービスや各種商品比較サービスなど、複数のウェブサイトの情報を取りまとめる仲介サービスを行うポータルサイトが次々と登場し、非常に注目を集めているわけだが、このような仲介サービスは、さらに画像の収集やMP3の収集など機能の専門化・多様化への発展を見せている。そのタスクの本質は、分散したウェブリソースを収集して加工した結果をウェブページとして提供するウェブサイト間のコンテンツ連携である。

## 【0015】

HTML技術では、ハイパーリンク機構を用いることにより任意のウェブページへジャンプできるようにしたり、フレーム機構を用いることにより複数のウェブページ全体を独立したウィンドウとして表示することはできるが、商品比較機能や合計値段見積み機能の提供といった有機的なコンテンツの連携を行うにはまったく不十分である。これらを実現するためには、任意のウェブページを収集して柔軟に加工する機能が必要である。HTMLのこのような機能欠如のため、CGI (Common Gateway Interface) やServlet

tなどのプログラム起動機構によって実行される外部プログラムやウェブサーバとは独立したデーモンプログラムにそれらの加工処理を行わせるという方法が取られている。この加工処理は概して次のような実行手続きが必要である。またデータベースを用いている場合は、さらにデータベースへのデータ登録や取出しの処理が加わる。

【 0 0 1 6 】

1. 外部ウェブサイトのHTMLページを取得する処理
2. HTMLページから必要なテキストを抽出する処理
3. 抽出されたテキストを所望の形式に変換する処理
4. テキストをつなぎ合わせて1つのHTMLを作成する処理

このような解決手法には欠点がある。すなわち、これらの処理の多くは仲介サービス間で内容的に似通っているにもかかわらず、それぞれサイト構築者が1からプログラムを作成しているというのは生産効率および保守性が悪い。また、作成されたプログラムはそのサイトの環境に依存するものであり、必然的にそのサイト専用のプログラム資産となってしまうため、他のサイト環境において再利用することが出来ない。

【 0 0 1 7 】

このような欠点は、WWW技術においてコンテンツ連携をターゲットに置き、それを容易に実現するためのツールあるいはシステムが存在しないことが原因である。

【 0 0 1 8 】

【発明が解決しようとする課題】

このように、従来は、複数のウェブページから必要とする情報を収集して、それを特定の書式に変換するといった加工を行った後、1つのウェブページ上に合成するための汎用的な手法がないという問題点があった。

【 0 0 1 9 】

今後、複数のウェブサイトの情報をとりまとめるポータルサイトのような仲介サービスがより活発化する状況下において、コンテンツ連携に特化した共通のプラットフォームを提供することは、生産効率およびポータビリティの面で有効な

手段の1つである。

【0020】

そこで、本発明は、上記問題点に鑑み、複数のウェブサイトの情報を1つのウェブ文書上に合成することが容易にしかも汎用的に行える文書合成方法およびそれを用いた文書合成装置を提供することを目的とする。

【0021】

【課題を解決するための手段】

本発明は、インターネットにおけるWWW (World Wide web) 上のマークアップ言語で記述された複数の第1の文書の内容の一部をWWW上のマークアップ言語で記述された第2の文書に合成するためのものであって、前記第1の文書の該インターネット上の所在と、該第1の文書から抽出する部分文書の範囲と、前記第2の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第2の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第2の文書に従って、前記第1の文書から前記部分文書を抽出して、その部分文書を前記第2の文書上の前記指定された挿入位置に挿入するとともに、前記変換ルールを用いて前記第2の文書上の前記指定された範囲の文書構造を変換することを特徴とする。

【0022】

本発明によれば、複数のウェブサイトの情報を1つのウェブ文書上に合成することが容易にしかも汎用的に行える。

【0023】

好ましくは、前記第2の文書は、前記第2の文書上の前記部分文書の挿入位置とを指定するとともに、前記第1の文書の所在と、該第1の文書から抽出する部分文書の範囲とを記述するため第1のタグ（挿入命令タグ `pz:targets`）と、前記変換ルールを用いて文書構造を変換すべき範囲を指定するとともに、前記変換ルールを記述したファイルの識別情報を記述するための第2のタグ（変換命令タグ `pz:convert`）とを用いて記述されている。

【0024】

また、好ましくは、前記第2の文書は、XML (E x t e n s i b l e M a r k u p L a n g u a g e) で記述されている。

【0025】

さらに、好ましくは、前記第1の文書がXMLで記述されていないときは、まず、XMLによる記述型式に変換した後、前記第1の文書から前記部分文書を抽出して、その部分文書を前記第2の文書上の前記指定された挿入位置に挿入する。

【0026】

なお、上記手法をインターネット上のウェブサーバに組み込み、クライアント装置（ウェブブラウザ）から前記第2の文書の要求を受けたとき、この第2の文書にの記述に従って1または複数の部分文書を合成した第2の文書を要求元のウェブブラウザに提供するサーバ装置を構成することができる。

【0027】

【発明の実施の形態】

以下、本発明の実施形態について図面を参照して説明する。

【0028】

なお、以下の説明は、次に示す項目の順になされている。

【0029】

(A) 複数のウェブサイトの情報を1つのウェブ文書に合成するために必要とされる機能

(B) XML-P' z 文書

(B-1) XML-P' z 言語の仕様

(B-2) XML-P' z 言語処理系の構成および動作

(C) 複数のウェブ文書を1つのウェブ文書上に合成するための一連の動作

(D) ウェブ文書の合成処理のためのXML-P' z サーバ間の協調動作

(E) 追記

(A) 複数のウェブサイトの情報を1つのウェブ文書に合成するために必要とされる機能

まず、実施形態の説明する前に、複数のウェブサイトの情報（ウェブ文書）を 1 つのウェブ文書に合成するために必要とされる機能について説明する。

#### 【0030】

複数のウェブ文書を 1 つのウェブ文書上に合成するために必要な機能は、抽出・挿入・変換の 3 種類に絞り込まれる。ただし、ウェブサイトの情報、すなわち、コンテンツとしてのウェブ文書（例えば HTML 文書）の全てが必要となるわけではなく、そのうちの一部のみが必要となるのが一般であることから、抽出機能には任意のウェブ文書のうちの部分文書を取り込むことが要求される。また、抽出された複数の部分文書を組み合わせて合成する際に、たとえば表の中に表を入れるというような柔軟な挿入機能が要求される。さらにそれだけでは不十分で、抽出してきた部分文書を一覧表型式に合成する際に、形式が不均一である場合に、それらを同じ形式に合わせるというように、文書の変換機能が要求されることもある。

#### 【0031】

この分析に基づき、本発明は、次のような記述モデルを採用する。まず、SSI (Server Side Inclusion) およびその発展系である ASP (Active Server Pages) や JSP (Java Server Pages) と同じように、複数のウェブ文書（部分文書）を合成するための合成用ウェブ文書内の任意位置にコマンドを配置し、そのコマンド実行結果が当該位置に埋め込まれるという、パッチワーク的な文書処理方式を採用する。

#### 【0032】

そして、用意するコマンドとして、どのウェブページのどの部分を抽出してどこに挿入するのかを示す部分文書の挿入コマンドを用意する。この方法は、抽出される部分文書の指定とその挿入位置を骨格となる合成用ウェブ文書を用いて自由にそして感覚的に記述できる利点がある。それに加えて、骨格となる合成用ウェブ文書の任意の範囲に対して、変換処理を施すことができる変換コマンドを用意する。この変換コマンドは、範囲情報と変換ルールを入力とし変換結果の文書を出力とする。まとめると、合成用ウェブ文書内の任意の位置に合成ロジックを

埋め込むことが出来る記述形式を採用し、合成ロジック用コマンドとして挿入および変換を用意した。

## 【0033】

また、採用した実行モデルの1つはSSIと同様であり、この合成用ウェブ文書をウェブサーバに配置しておき、ブラウザからそのURLへの要求があった場合に、そのウェブサーバに配置された言語処理系がその合成用ウェブ文書に含まれるコマンドを解釈実行し、その結果をブラウザに返すというものである。この方法では、サイト構築者は、合成用ウェブ文書をウェブサーバに配置しておくだけで解釈実行の起動について意識しなくてよいという利点がある。ただし、そのような実行方法だけではなく、ユーザが手動で解釈実行を行わせることも原理的に可能である。この場合、クライアント側で任意の合成を行うことができる。

## 【0034】

さて、このような合成用ウェブ文書の記述においてXML (Extensible Markup Language) は最適な言語である。XMLはタグ名や属性名を自由に定義し、それに対してアプリケーション側がセマンティクスを与えることが出来る。それに加えて、またXMLはツリー型の文書構造を持つことが保証されているため、ツリー構造で表現される文書構造上における1つのノードとして表される特定のエレメントを指し示すだけで部分文書(文書範囲)を指定することができる。

## 【0035】

また、XML自体はローレベルでの標準のデータ形式としての需要から、XSLT (Extensible Stylesheet Language Transformations) (参考文献: <http://www.w3.org/TR/xslt>) などの変換系技術も整備されているし、今後のXML技術の発展においても上記の合成用ウェブ文書を、このXML言語を応用した言語(本発明に係るXML応用言語)で記述することで拡張性およびツール利用などの利便性が約束されることになる。

## 【0036】

また、将来、HTML文書だけでなくXML文書がよく用いられるようになっ

たときにも、抽出対象として扱いやすいという利点がある。

【 0 0 3 7 】

そこで、本発明では、合成用ウェブ文書の記述言語をXML応用言語として具体的に設計する。

【 0 0 3 8 】

本発明では、結合のためのベースとなる合成用ウェブ文書（合成用ウェブページと呼ぶこともある）をXMLで記述し、指定した他のウェブ文書から指定した範囲の部分（部分文書）を抽出して、それを合成用ウェブ文書の指定された位置に挿入し、合成用ウェブ文書の指定した範囲に変換処理（所望の文書構造への変換処理）を施す、挿入・変換の2つの合成ロジック命令をその合成用ウェブ文書内にエレメントとして持たせる方針を採る。

【 0 0 3 9 】

このような合成用ウェブ文書、すなわち、XML文書（XMLページ）を、ここでは、XML-P' z. (XML-P i e c e s) 文書（XML-P' z ページ）と呼ぶものとする。

【 0 0 4 0 】

XML-P' z 言語処理系をウェブサーバへ組み込みむことにより、図1に示すような動作が可能になる。なお、XML-P' z 言語処理系を組み込んだウェブサーバをXML-P' z サーバと呼ぶこともある。具体的には、Microsoft社のウェブサーバであるIIS (Internet Information Server) への組み込む場合を例にとり説明する。

【 0 0 4 1 】

図1に示した基本的な動作原理において、  
 （ステップS101）クライアント端末B1のウェブブラウザからXML-P' z サーバA1（以下、簡単にサーバA1と呼ぶ）へのXML-P' z 文書2の要求（GET/HTTP）が送信される。

【 0 0 4 2 】

（ステップS102）サーバA1は、要求されたリソースがXML-P' z 文書かどうかを判断する。



## 【0043】

(ステップS103) XML-P' z 文書と判断した場合、サーバA1は、XML-P' z 言語処理系(図1の合成処理部1)を起動し、XML-P' z 文書2に記述されている、指定されたウェブサーバ(例えば、ここでは、ウェブサーバA2、A3)のウェブ文書(ページ)W2、W3から指定した範囲の部分(部分文書)を抽出し、それをXML-P' z 文書の指定位置に挿入するとともに、XML-P' z 文書に記述されている指定された範囲に変換処理を施す。最終的に、XML-P' z 言語処理系の処理結果としてのXML文書(合成されたウェブ文書)W1を得る。

## 【0044】

(ステップS104) 得られたXML文書を要求元への返答としてブラウザに送信する。

## 【0045】

上記動作は、ウェブサーバの設定によって実現する。ほとんどのウェブサーバには、URL文字列のパターン(よくあるのがオブジェクトの拡張子)とそれを前処理するのに必要なアドインを対応付ける機能を持っており、それを利用することにより(ステップS102)～(ステップS103)を実現できる。

## 【0046】

また、ウェブブラウザがXML文書を表示できる場合はXML文書を、表示できない場合はサーバA1側でスタイルシートを処理してHTML文書を返すという処理があってもよい。

## 【0047】

## (B) XML-P' z 文書

XML-P' z 文書では、挿入命令エレメント「pz:targets」と変換命令エレメント「pz:convert」とを定義する。

## 【0048】

挿入命令タグを用いることにより、XML-P' z 文書のツリー構造で表現される文書構造上における1つのエレメント下の子文書として他のXML文書また

はHTML文書の部分文書を挿入（合成）することができる。挿入対象とする部分文書の指定としては、XPointer付URL（参考文献：<http://www.w3.org/TR/WD-xptr#uri-escaping>）を採用する。これにより1行で簡潔に特定ウェブページの部分文書を指定することが出来る。ただしXPointer規格はXMLのためのものであるため、HTMLを直接対象とすることが出来ない。このことから、抽出する際に、HTML-DOM（Document Object Model）およびXML-DOMを用いることにより、構造的に等価なHTML-XML変換を行う機構を導入する。これによりHTML文書はXML文書として扱うことが出来るので、すべての加工処理はXMLとして行うことが出来るようになる。

## 【0049】

またXML-P'z文書では、変換命令エレメントを用いることにより、任意のエレメント（ノード）下の各子文書に対してXSLT（Extensible Style Language transformations）を用いた変換操作を実行することができる。すなわち、変換命令エレメントによって指示された、変換命令エレメントの子ノードとして配置される各子文書に対して指定されたXSLTが適用される。これを利用して、挿入命令タグによって挿入されたウェブ文書を変換命令タグを用いて変換することができる。

## 【0050】

以下は、挿入命令エレメントと変換命令エレメントとを用いた、挿入機能と変換機能を有するXML-P'z文書の単純な例である。

## 【0051】

（XML-P'z文書の第1の例）

1. `<?xml version="1.0" ?>`
2. `<root xmlns:pz="http://www.shiba.co.jp/xmlpz" >`
3. `<category>xxx</category>`
4. `<item_holder>`
5. `<pz:convert href="xxx.xsl" >`
6. `<pz:targets href="http://www.yyy.com/index.xml#xpointer(//item)`

" />

7. </pz:convert>

8. </item\_holder>

9. </root>

図11(a)は、上記第1の例の文書構造を模式的に示したもので、図11(b)は、上記第1の例を解釈した後のXML文書の文書構造を模式的に示したものである。

#### 【0052】

上記第1の例において、6行目の挿入命令エレメント「p z : t a r g e t s」で指定された挿入対象の各XML部分文書（http://www.yyy.com/index.xml#xpointer(//item)で、以下、簡単に部分文書PD1と呼ぶ）が、5行目の変換命令エレメント「p z : c o n v e r t」で指定されたXSLTの変換ルールが適用されて変換され、4行目～8行目にある「i t e m \_ h o l d e r」エレメントの子エレメントとして、図11(b)に示すように、挿入される。ただし、6行目の「p z : t a r g e t s」で指定されているウェブ文書はXPointerにマッチするすべての部分文書であり（上記第1の例の場合は、「i t e m」タグがルートとなる部分文書すべて）、一般的には複数のウェブ文書となる。

#### 【0053】

上記の分散ウェブリソースのウェブ文書合成手法は以下の優位性がある。

#### 【0054】

優位点の一つは構築容易性である。本手法は、データベースを中心とした従来の方式と異なり、情報資源の合成ロジックをプログラミング言語なしで簡潔に記述できるので、ウェブ文書統合の構築・構成変更が容易である。またブラウザからの要求時に解釈処理されるインタプリタ型の実行モデルが採用されているので、合成ロジックの変更はただちに反映される。

#### 【0055】

もう一つの優位点は高い再利用性にある。XML-P'zのフレームワークでは、コンテンツ・変換ルール・合成ロジックなどすべての構成要素がウェブリソ

ースとして提供される。ウェブ文書の外にプログラムとして合成ロジックを持たせていた従来の方法と異なり、本方式ではURLを介してこれらすべての構成要素にアクセスすることができるので、原理的に世界中のウェブシステムから再利用することができる。このことはウェブサイトを越えた分散システムに必要な各リソースを自由に配置することを意味し、運用に応じた柔軟なシステム構築および変更が可能となる。

## 【0056】

さらにXML-P'z文書が別サイトのXML-P'z文書を合成対象とすることでウェブサイト間で合成ロジックを分業（連携）することができる。

## 【0057】

またHTTP以外の特別なプロトコルをまったく用いておらず、ウェブリソースを提供する側ウェブサイトは特別な処理システムを導入する必要がない。したがってあらゆるウェブサイトの情報資源を再利用対象とすることができる。言い換えれば、既存のウェブサイトはシステム資源をそのまま生かすことが出来、XML-P'z資源を別途作成するだけで合成することが出来る。

## 【0058】

ただし、このような高いアクセシビリティについては、著作権問題など利用に関する実運用上の問題がからむ。たとえば、XML-P'z技術を用いれば、ウェブ検索サービスを行っている複数のウェブサイトの検索結果を合成するメタ検索ページを提供することが簡単にできるが、著作権問題に抵触する。このような問題は、現在のWWWにおいてもハイパーリンクの許可をめぐる問題となっており運用で乗り切っている現状がある。これに対して、Extranet構築技術などアクセスコントロールに関するWWW技術が提供されている一方、WWWで公開された著作物の取り扱いに関する法整備が急ピッチで行われているところである。またXML-P'zフレームワークにおいても、将来の課題として著作権問題を包括的に取り扱うモデルを導入したいと考えている。

## 【0059】

次に、以上、説明した分散ウェブリソースのウェブ文書合成手法を次の2つのパートに分けて説明する。

## 【 0 0 6 0 】

(B-1) XML-P' z 言語の仕様

(B-2) XML-P' z 言語処理系の構成および動作

XML-P' z 言語とは、合成ロジックを含むウェブページ記述言語であり本システムの中核をなす。まずその言語仕様について (B-1) で説明する。次に XML-P' z 言語で記述された XML-P' z 文書を解釈処理し、その結果を返す言語エンジンとしての言語処理系の構成およびその動作について (B-2) で説明する。

## 【 0 0 6 1 】

(B-1) XML-P' z 言語の仕様

XML-P' z 言語とは、特定のタグ名に対してセマンティクスが与えられた XML 応用言語の 1 つであり、分散ウェブリソースの合成を目的としたウェブ文書記述言語である。通常の XML 文書と同様、コンテンツを記述することができるのに加え、任意の要素に対して、ウェブリソースを操作する命令用のタグ名を記述することにより、合成ロジックを内部に含めることができる。この合成ロジックの記述は HTML のハイパーリンクのように簡潔である。

## 【 0 0 6 2 】

このように合成ロジックを含む XML-P' z 言語にて記述された XML-P' z 文書は、その合成ロジックに従い仮想的に分散リソースを統合・合成したウェブ文書へと解釈される。

## 【 0 0 6 3 】

ウェブリソース操作に関する命令要素として「targets」および「convert」の 2 つが用意されており、XML ネームスペースとして「pz」を予約している。これらの命令要素を組み合わせ用いることにより、他のウェブ文書を含めた任意の部分文書の抽出および自文書の挿入や XSLT を用いた構造変換を行うことができる。以下に各命令要素 (pz:convert 要素、pz:targets 要素) について説明する。

## 【 0 0 6 4 】

また、これらの命令要素は深さ優先の探索順序で解釈されなければなら

ない。たとえば、図12に示すXML-P' z文書の文書構造において、pz: convert要素の子要素として、pz: targets要素が複数ある場合、各pz: targets要素が兄から弟へ順に解釈された後、pz: convert要素が解釈される。

## 【0065】

また、各命令タグの項でも説明しているとおり、挿入命令要素によって挿入されるウェブ文書および変換命令要素によって変換するウェブ文書は、合成、変換する前にXML-P' z文書として解釈されなければならない。すなわち、命令要素によって挿入、変換するウェブ文書内に命令要素（挿入、変換命令要素）が含まれている場合、それらが優先的に上述の順序で解釈されたのち、挿入先である本XML-P' z文書の解釈実行が続行されるという再帰的な解釈処理の流れとなる。

## 【0066】

また、ウェブリソースの指定子としてXPointer付URLを導入している。これはXPointer規格（参考文献：<http://www.w3.org/TR/WD-xptr>）に準拠するものであるが、本規格ではXPointer付URLの相対指定について未定義であるので、XML-P' z言語では独自に規格を定めている。

## 【0067】

以下にその規格を示す。

## 【0068】

（XMLネームスペース）

XML-P' zの各命令タグを利用するためには、以下のネームスペースを宣言しなければならない。

## 【0069】

・ネームスペース名

pz

・ネームスペースURI

`http://shiba.co.jp/xmlpz`

(`pz:targets` エレメント)

任意のウェブリソースを抽出・挿入する

文法

```
<pz:targets
href="web-resources-url">
</pz:targets>
```

・属性

`href`

挿入対象となる複数のウェブリソースへのURL。URLがXPointer付である場合、URLのボディ部のウェブ文書においてXPointerパターンにマッチするすべての部分文書が指定される。

【0070】

・構造制約

親エレメント： 任意

子エレメント： なし

・注釈

`pz:targets` エレメントは、`href` 属性によって指定された単数あるいは複数のウェブリソースをXML-P' z文書として解釈したのち当該エレメントのコンテキストに対して挿入し、`pz:targets` エレメント自身は消滅する。`href` 属性によって示されるURLがXPointer付である場合、URLのボディ部のウェブ文書においてXPointerパターンにマッチするすべての部分文書が指定される。

【0071】

・サンプル

以下の例は、自文書内に含まれている本のデータに加え、「`http://www.xxx.com/booklist.xml`」ページ内に含まれる本データをすべて取り込むXML-P' z文書である。

【0072】

```

1. <?xml version=" 1.0" ?>
2. <bookstore specialty=" novel"
3.           xmlns:pz=" http://www.shiba.co.jp/xmlpz" >
4. <book style=" textbook" >
5.   <author>
6.     <first-name>Shinichiro</first-name>
7.     <last-name>Hamada</last-name>
8.     <publication>Selected Short Stories of
9.       <first-name>Shinichiro</first-name>
10.      <last-name>Hamada</last-name>
11.    </publication>
12.  </author>
13.  <price>55</price>
14. </book>
15. <pz:targets href=" http://www.xxx.com/booklist.xml#xpointer(//boo
k)" />
16. </bookstore>

```

(pz:convert エレメント)

任意の部分文書群を XSLT 文書を用いて変換する  
文法

```

<pz:convert
href=" xslt-url" >
</pz:targets>

```

属性

href

変換ルールを定義する XSLT 文書への URL。URL が XPointer 付である場合、URL のボディ部のウェブ文書において XPointer パタ



ーンにマッチする部分文書のうち、文書順で先頭の部分文書が指定される。

【0073】

構造制約

親エレメント： 任意

子エレメント： 任意

注釈

pz:convertエレメントは、当該エレメント下の各子文書それぞれに対して、href属性によって指定されたXSLT文書を適用して変換する。変換された各子文書は、XML-P'z文書として解釈した後pz:convertエレメントのコンテキストに挿入され、pz:convertエレメント自身は消滅する。href属性によって示されるURLがXPointer付である場合、URLのボディ部のウェブ文書においてXPointerパターンにマッチする部分文書のうち、文書順で先頭の部分文書が指定される。

【0074】

サンプル

以下の例は、「textbook」エレメントで表現されている自文書内に含まれている教科書データに加え、「http://www.xxx.com/booklist.xml」ページ内に含まれるすべての教科書データを「textbook-book.xsl」というXSLT文書に記述された変換ルールに従って、共通書籍形式へ変換し、また、「http://www.yyy.com/index.html」ページで公開されている本データを共通書籍形式へ変換したものをすべて取り込むXML-P'z文書である。

【0075】

1. <?xml version="1.0" ?>
2. <bookstore specialty="novel"
 

xmlns:pz="http://www.shiba.co.jp/xmlpz">
3. <pz:convert href="textbook-book.xsl">
4. <textbook>
5. <author>

```

6.      <first-name>Shinichiro</first-name>
7.      <last-name>Hamada</last-name>
8.      <publication>Selected Short Stories of
9.      <first-name>Shinichiro</first-name>
10.     <last-name>Hamada</last-name>
11.     </publication>
12.     </author>
13.     <price>55</price>
14.     </textbook>
15.     <pz:targets href=" http://www.xxx.com/booklist.xml#xpointer(//
textbook)" />
16.     </pz:convert>
17.     <pz:convert href=" html-book.xml" >
18.     <pz:targets href=" http://www.yyy.com/index.html#xpointer(//TA
BLE[2] //TR)" />
19.     </pz:convert>
20. </bookstore>

```

#### (X P o i n t e r 付URLの相対指定)

ウェブリソースが他のウェブリソースを参照指定する際に、自ウェブリソースの持つURLをベースとして相対的なURLを用いることができる。これを相対URLと言う。資源を一意に区別するためには、処理系が相対URLを絶対URLへ展開しなければならない。その解決方法を以下に示す。ただし以下の説明において、用語はIETF (<http://www.ietf.org/rfc/rfc1738.txt>) に基づくものとする。

#### 【0076】

1.) ベースURLのオブジェクトと相対URLのオブジェクトが異なる場合  
 ベースURLから (もしあれば) X P o i n t e r フラグメントを取り除いたボディ部と、相対URLから (もしあれば) X P o i n t e r フラグメントを

取り除いたボディ部との間で、IETF (<http://www.ietf.org/rfc/rfc1808.txt>) に基づいた相対URLの解決を行った結果に対して、(もしあれば) 相対URLのXPointerフラグメントを与える。なお、XPointerフラグメントとは、例えば、以下のサンプルの記述における「#xpointer」以下の部分で、「#xpointer(/node1/node2)」や、「#xpointer(. /node3//node4)」である。

【0077】

・サンプル

(ベースURL) [http://aaa.com/dir1/xxx.xml#xpointer\(/node1/node2\)](http://aaa.com/dir1/xxx.xml#xpointer(/node1/node2))

(相対URL) [. /dir2/yyy.xml#xpointer\(. /node3//node4\)](http://aaa.com/dir1/xxx.xml#xpointer(. /node3//node4))

(解決結果) [http://aaa.com/dir1/dir2/yyy.xml#xpointer\(. /node3//node4\)](http://aaa.com/dir1/dir2/yyy.xml#xpointer(. /node3//node4))

2.) ベースURLのオブジェクトと相対URLのオブジェクトが同じ場合

ベースURLがXPointerフラグメントを含んでいる場合はXPointerが示す文書ノード、XPointerフラグメントを含んでいない場合はルート文書ノードを起点として、(もしあれば) 相対URLのXPointerの示すノードを決定し、そのノードパスを示すXPointerフラグメントを当該オブジェクトのURLに与える。

【0078】

・サンプル

(ベースURL) [http://aaa.com/dir1/xxx.xml#xpointer\(/node1/node2\)](http://aaa.com/dir1/xxx.xml#xpointer(/node1/node2))

(相対URL) [http://aaa.com/dir1/xxx.xml#xpointer\(. /node3//node4\)](http://aaa.com/dir1/xxx.xml#xpointer(. /node3//node4))

(解決結果) [http://aaa.com/dir1/xxx.xml#xpointer\(/node1/node2/node3//node4\)](http://aaa.com/dir1/xxx.xml#xpointer(/node1/node2/node3//node4))

## 3.) 相対URLにおいてオブジェクトが無指定である場合

ベースURLがXPointerフラグメントを含んでいる場合はXPointerが示す文書ノード、XPointerフラグメントを含んでいない場合はルート文書ノードを起点として、(もしあれば)相対URLのXPointerの示すノードを決定し、そのノードパスを示すXPointerフラグメントをベースURLのオブジェクトのURLに与える。

【0079】

## サンプル

(ベースURL) `http://aaa.com/dir1/xxx.xml#xpointer(/node1/node2)`  
 (相対URL) `#xpointer(. /node3//node4)`  
 (解決結果) `http://aaa.com/dir1/xxx.xml#xpointer(/node1/node2/node3//node4)`

## (B-2) XML-P' z 言語処理系の構成および動作

次に、XML-P' z 言語の解釈処理系について説明する。

【0080】

XML-P' z 言語処理系は、XML-P' z 文書の所在を示すURLまたはソースを入力とし、その解釈結果のXML文書ソースを出力とするソフトウェアコンポーネントである。本処理系ではXML-P' z 言語の解釈処理を2パスで行う方式を取っており、1パス目でXMLとして構文解析を行ってXML-DOMツリーを作成し、続いて2パス目でXML-DOMツリーを深さ優先でたどりながら、XML-P' z 言語特有の命令エレメント(挿入、変換命令タグで囲まれた部分)の解釈処理を行う。この言語処理に際して、文法逸脱を発見した場合やネットワークトラブルなどのランタイムエラーが発生した場合でも、解釈処理をそのまま続行することにより、可能な最良の結果を出力する処理方針をとる。

【0081】

またXML-P' z 言語ではXPointer付URLを用いたウェブリソース指定が可能であるが、本処理系では、URLで示される文書全体をダウンロード

ドした上で、X P o i n t e r で指定された部分文書を切り出すという２段階の処理を行う方式を取る。これにより、X P o i n t e r 付URLに対応していないほとんどのウェブサーバに対しても、ウェブリソースを要求することが出来る。

#### 【 0 0 8 2 】

以上が基本的な処理方針である。この処理方針に基づいた本処理系のシステム構成例について説明する。

#### 【 0 0 8 3 】

図 2 は、XML - P' z 言語処理系 1 0 0 (図 1 の合成処理部 1 に相当) の全体の構成例である。図 2 において、この言語処理系 1 0 0 は、大きく分けて、XML - P' z 文書読込に関する処理モジュールである、解釈バッファファクトリ 1 0 1 と、読み込まれた文書を解釈した結果のXMLを返す処理モジュールである、インタプリタ 1 0 2 の 2 つから構成されている。これらは基本的に独立に動作する。なお、図 2 中の 2 つの解釈バッファファクトリ 1 0 1 は同一物であるが見やすくするため分けて書いている。

#### 【 0 0 8 4 】

解釈バッファファクトリ 1 0 1 は、XML - P' z 文書の所在を示すURLまたはソースの入力をトリガとして動作を開始し、まず、XML ノーマライザ 1 1 1 において、入力文書がXMLならばそのまま、HTMLならば同等の構造を持つXMLへの等価変換処理を行った上で、XML - DOM パーサ 1 1 4 を用いてXML - DOM ツリーを作成し、さらに、X P o i n t e r プロセッサ 1 1 5 において、URL 内に含まれるX P o i n t e r フラグメントにしたがって部分文書を抽出した結果をもとに、解釈バッファイニシャライザ 1 1 6 は、解釈バッファ 1 0 3, 1 0 4 を生成する。

#### 【 0 0 8 5 】

さらに、URL またはソースの入力が処理系 1 0 0 外部からであった場合、生成する解釈バッファを、デフォルト解釈バッファ 1 0 3 として登録する。ここで解釈バッファとはXML - P' z 言語解釈処理の状態記憶でありインタプリタ 1 0 2 の解釈処理中に繁茂に更新される。

## 【0086】

一方、インタプリタ102は処理系100外部からの解釈結果の要求があった場合に動作を開始し、デフォルト解釈バッファ103の解釈用XML-DOMツリー131を深さ優先でたどりながら、pz:targetsエレメントおよびpz:convertエレメントの2つの命令エレメントの解釈実行を行い、最終的に得られた解釈結果のXML文書を出力する。

## 【0087】

ただし、命令エレメントの解釈中に一時的に生成される部分文書をXML-Pz解釈処理するため、解釈バッファファクトリ101を用いて、一時解釈バッファ104を生成する。

## 【0088】

次に、解釈バッファファクトリ101を構成する各構成部（モジュール）の処理動作を説明する。

## 【0089】

解釈バッファファクトリ101を構成する、XMLノーマライザ111は、HTML判定器112、および、HTML-XMLコンバータ113から構成される。

## 【0090】

HTML判定器112は、与えられたURLが指し示すウェブリソース（ウェブ文書）がHTML文書かXML文書かを判定する。その判定にはHTTPヘッダの「Content-type」を用いる方法とURL内に含まれる拡張子を用いる方法の2段階のテストを行う。この処理動作を図3に示す。

## 【0091】

図3において、まず、「Content-Type」を取得する（ステップS1）。この取得の方法として当該URLに対して、HEAD要求を行うのがもっとも直接的である。しかしHEAD要求を理解できないウェブサーバも世の中にたくさんある。代用としてGET要求を用いることもできる。次に、当該URLに対してHTTP接続できたかどうか判定する（ステップS2）。もし接続に成功した場合は、ステップS3へ進み、失敗した場合はステップS5に進む。

## 【0092】

ステップS3では、「Content-Type」ヘッダを取り出し、その中に「text/html」という文字列が含まれているか判定する。もし含まれていればHTMLと判定して終了し（ステップS6）、そうでなければ、XMLと仮判定して終了する（ステップS4）。

## 【0093】

ステップS5では、URL内のオブジェクトフィールドの拡張子が「html」または「htm」であるかどうか判定する。もしそうであればHTMLと判定して終了し（ステップS6）、そうでなければXMLと仮判定して終了する（ステップS7）。

## 【0094】

HTML-XMLコンバータ113は、HTML判定器112によってHTML文書と判断されたウェブリソースを構造的に等価なXML文書へ変換する。これはHTML-DOMツリーからXML-DOMツリーへと各DOMのメソッドを用いて順次移していくことで実現できる。HTML-XMLコンバータ113の処理動作を図4に示す。

## 【0095】

まず、ステップS11において、与えられたHTML文書をHTMLパーサへ読み込ませ、HTML-DOMツリーを構築する。HTMLパーサはウェブブラウザが内部的に用いているものが望ましい。なぜならウェブブラウザが使用するHTMLパーサは、HTML文法逸脱に対するエラーリカバリー機能がついているからである。

## 【0096】

次に、ステップS12において、XML-DOMパーサを用いて空のXML-DOMツリーを構築する。そして、ステップS13において、HTML-DOMツリーを全探索しながら、立ち寄ったノードの値などを取り出しXML-DOMツリーにノードとして挿入する。

## 【0097】

以上の処理により、XMLノーマライザ111は、解釈バッファファクトリ1

01にURLとして入力されたウェブリソースをすべてXML文書として出力する。一方、ソースとして入力されたウェブリソースはすべてXML文書と仮定して取り扱われる。

【0098】

XMLノーマライザ111を通過したXML文書またはソースとして入力されたXML文書は、XML-DOMパーサ114に入力され、XML-DOMツリー化される。さらに、XPointerプロセッサ115を用いて、URLのXPointerフラグメントで示されているXML文書内の部分文書のXML-DOMツリーを得る。XPointerプロセッサ115のXPointerフラグメントに対する処理動作を図5に示す。

【0099】

まず、ステップS21で、与えられたウェブリソースがURLによるものだったのか、ソースによるものだったのかを判定する。ソースによるものであった場合URLは存在しないので、この時点で終了する。

【0100】

次に、ステップS22において、URLのフラグメントからXPointerフラグメントを取り出す。ただしXPointerが指定されていなかった場合は空の文字列とする。続いて、ステップS23においてXML-DOMツリーのルートエレメントを基点としてXPointerが指し示すノードを同定する。これには一般的なXPointer処理系を用いればよい。

【0101】

次に、ステップS24において指し示されたノードがエレメントであるかどうかを判定する。もしエレメントでなければ異常終了する。続いて、ステップS25において、得られたエレメントをルートエレメントとした部分文書のXML-DOMツリーを切り出す。さらに、ステップS26において、その切り出されたXML-DOMツリーを新しいXML文書のXML-DOMツリーとする。

【0102】

さて、得られたXML-DOMツリーを基に、解釈バッファイニシャライザ116は解釈バッファを生成する。このとき与えられたウェブリソースが言語処理



系 1 0 0 外部からの入力によるものであった場合、その解釈バッファを、デフォルト解釈バッファ 1 0 3 として登録する。この解釈バッファ（メモリで構成されている）の初期化処理動作を図 6 に示す。なお、部分文書の XML-DOM ツリーの場合は、一時解釈バッファ 1 0 4 を図 6 と同様にして初期化する。

#### 【 0 1 0 3 】

まず、ステップ S 3 1 では、与えられた XML-DOM ツリーをソース XML-DOM ツリー 1 3 4 にコピーする。なお、ソース XML-DOM ツリー 1 3 4 は、以後の XML-P' z 言語の解釈処理によって変更される前の XML-DOM ツリーの初期状態を記憶するバッファであり、XML-P' z 言語のソース提供などの用途を想定しているが、本実施形態では利用されない。

#### 【 0 1 0 4 】

次に、ステップ S 3 2 では、与えられた XML-DOM ツリーを解釈用 XML-DOM ツリー 1 3 1 へコピーする。解釈用 XML-DOM ツリー 1 3 1 は、インタプリタ 1 0 2 が解釈処理において構造の読み込みおよび解釈結果の書き込みに用いる。

#### 【 0 1 0 5 】

ステップ S 3 3 では、プログラムカウンタ 1 3 2 を解釈用 XML-DOM ツリー 1 3 1 のルートエレメントにセットする。プログラムカウンタ 1 3 2 は、インタプリタ 1 0 2 の解釈処理の進捗を記憶するポインタである。

#### 【 0 1 0 6 】

最後に、ステップ S 3 4 では、ロードフラグ 1 3 3 を「false」にセットする。ロードフラグ 1 3 3 とは、当該解釈バッファ 1 0 3 がすでに解釈処理済みかどうかを示すフラグである。インタプリタ 1 0 2 は、このフラグ 1 3 3 を利用して過去に解釈処理を施した解釈バッファについて解釈処理をし直さないようになっている。

#### 【 0 1 0 7 】

以上が、解釈バッファファクトリ 1 0 1 の処理動作の説明である。

#### 【 0 1 0 8 】

次に、インタプリタ 1 0 2 の処理動作について説明する。

## 【0109】

インタプリタ102を構成するコンテキストマネージャ121は、解釈処理において中心的役割を果たす。解釈バッファ103、104のプログラムカウンタ132、142に従い、解釈用XML-DOMツリー131、141の各ノードを深さ優先で立ち寄る際に、命令エレメントを発見すると該当する処理モジュール(`targets`コマンドプロセッサ122、`convert`コマンドプロセッサ123)へ解釈処理を依頼する。命令エレメントの解釈処理が終了すると立ち寄り処理を続行する。すべての処理が終わると解釈結果としてXML文書出力する。この処理動作を図7に示す。以下、デフォルト解釈バッファ103を用いた解釈処理の場合を説明するが、一時解釈バッファ104の場合も同様である。

## 【0110】

まず、ステップS41において、解釈バッファ103のロードフラグ133を調べる。ロードフラグが「`true`」であればすでに解釈済みであり「`false`」ならば、まだ解釈処理が行われていない状態であることを意味する。「`true`」ならば、ステップS49へ進み、「`false`」ならば、ステップS42へ進む。

## 【0111】

ステップS42では、プログラムカウンタ132を読み込んで解釈処理対象とするエレメント(これをカレントエレメントと呼ぶ)を決定する。

## 【0112】

ステップS43では、カレントエレメントのエレメント名が「`pz:targets`」かどうかをチェックし、「`pz:targets`」だった場合は、ステップS4へ進み、`pz:targets`エレメントの解釈処理を`targets`コマンドプロセッサ122へ依頼する。

## 【0113】

続いて、ステップS45では、カレントエレメントのエレメント名が「`pz:convert`」かどうかチェックし、「`pz:convert`」だった場合は、ステップS46へ進み、`pz:convert`エレメントの解釈処理を`con`

vertコマンドプロセッサ123へ依頼する。

【0114】

続いて、ステップS47で、深さ優先で移動先エレメントを決定しプログラムカウンタにセットする。カレントエレメントの子エレメントのうち、まだ解釈処理を行っていないエレメントがあれば、そのうちの長兄エレメントをプログラムカウンタへセットする。すべての子エレメントの解釈処理が行われているならば、親エレメントにプログラムカウンタへセットする。ただし親エレメントがない場合は、プログラムカウンタを「NULL」にセットする。

【0115】

ステップS8では、プログラムカウンタ132が「NULL」かどうかをチェックし、「NULL」でなければ、ステップS42へ戻る。「NULL」であれば、解釈用XML-DOMツリー131の解釈は終了したので、ステップS49へ進む。

【0116】

ステップS49では、XML-DOMパーサ151を用いて解釈バッファ103のXML-DOMツリー131を基にXML文書を生成し出力し、終了する。

【0117】

インタプリタ102を構成するtargetsコマンドプロセッサ122は、pz:targetsエレメントを解釈し、その結果をカレントエレメントに書き込む。この処理動作を図8に示す。

【0118】

まず、ステップS51では、カレントエレメントであるpz:targetsエレメントのhref属性値を取り出し、ステップS52で、その属性値を解釈バッファファクトリ101の入力URLとして、前述したXMLノーマライザ111から解釈バッファイニシャライザ116による処理を経由して、一時解釈バッファ104を生成する。ただし、対象とするURLが相対URLであった場合は、前述の「XPointer付URLの相対指定」の説明に基づき、挿入先の解釈バッファのURLをベースとして絶対URLへ変換する。

【0119】

次に、ステップS53へ進み、生成された一時解釈バッファ104を、インタプリタ102を用いて解釈処理し、その結果としてのXML文書を得る。

#### 【0120】

最後に、ステップS54では、DOMパーサ152を用いて、得られたXML文書をXML-DOMツリーに変換して、カレントエレメントである「pz:targets」エレメントと入れ替える。また、生成した一時解釈バッファ104は破棄する。

#### 【0121】

インタプリタ102を構成するconvertコマンドプロセッサ123は、convertエレメントを解釈し、その結果をカレントエレメントに書き込む。この処理動作を図9に示す。

#### 【0122】

まず、ステップS61では、カレントエレメントであるpz:convertエレメントのhref属性値を取り出し、ステップS62で、その属性値を解釈バッファファクトリ101の入力URLとして、前述したXMLノーマライザ111から解釈バッファイニシャライザ116による処理を経由して、一時解釈バッファ104を生成する。ただし、対象とするURLが相対URLであった場合は、前述の(XPointer付URLの相対指定)の説明に基づき、挿入先の解釈バッファのURLをベースとして絶対URLへ変換する。

#### 【0123】

次に、ステップS63へ進み、生成された一時解釈バッファ104を、インタプリタ102を用いて解釈処理し、その結果としてXSLT文書を得る。なお、このような処理を行うのは、XSLT文書自体がXML-P'z言語でかかっている可能性があるからである（すなわち合成結果としてXSLT文書が構成されている可能性があるからである）。

#### 【0124】

続いて、ステップS64へ進み、XSLTプロセッサ124により、カレントエレメントである「pz:convert」エレメントの子エレメントのうち、まだXSLTを適用していない長兄エレメント（およびその子孫エレメントを含

む部分文書)に、得られたXSLT文書を用いて、当該部分文書の文書構造をXSLT文書に記述された変換ルールを用いて変換し、その変換して得られたXML-DOMツリーを、ステップS65では、合成用ウェブ文書上の変換前の子エレメント(およびその子孫エレメントを含む部分文書)と入れ替える。

## 【0125】

ステップS66において、もし未処理の子エレメントがあるならば、ステップS64に戻る。すべての子エレメントが処理済ならば、ステップS67へ進み、`pz:convert`エレメントを`pz:convert`エレメントの各子部分文書である文書構造の変換されたものと入れ替える。

## 【0126】

以上が、インタプリタ102の処理動作であり、以上をもってXML-P'z言語処理系の各構成部についての説明は終了した。

## 【0127】

(C) 複数のウェブ文書を1つのウェブ文書上に合成するための一連の動作

次に、図2に示した構成のXML-P'z言語処理系100をウェブサーバへ組み込み、図1に示した基本的な動作を行って、実際に、ウェブサーバA2のウェブ文書W2からその一部を抽出し、その抽出された各部分文書を1つのウェブ文書上に合成し、合成されたウェブ文書(XML文書)W1を出力するための一連の動作を図13～図15に示すフローチャートを参照して説明する。

## 【0128】

ここで、合成用ウェブ文書としてのXML-P'z文書2は、図16に示すものであるとする。なお、図16に示すXML-P'z文書は、図1のXML-P'z文書2のうちの一部分を抜粋したものを示している。

## 【0129】

図16に示すXML-P'z文書は、「textbook」エレメントE1で表現されている自文書内に含まれている教科書データと、`pz:targets`エレメントE2にて挿入される「`http://www.xxx.com/booklist.xml`」のウェブ文書内に含まれるすべての教科書データとを、

「textbook-book.xml」というXSLT文書に記述された変換ルールに従って、共通書籍形式へ変換して、合成されたウェブ文書（XML文書）W1を出力するためのものである。

#### 【0130】

図1において、クライアント端末B1のウェブブラウザからXML-P'zサーバA1（以下、簡単にサーバA1と呼ぶ）へのXML-P'z文書2の要求がなされたとする（ステップS201）。

#### 【0131】

サーバA1の言語処理系100は、要求された文書が自身を持つ合成用ウェブ文書（XML-P'z文書）2であるので、XML-DOMパーサ114を用いて当該XML-P'z文書のXML-DOMツリーを作成する（ステップS202）。この作成されたXML-DOMツリーの図16に対応する部分は、例えば、図17に示すものである。なお、図17では、説明の簡単のために概略的に示している。

#### 【0132】

この作成されたXML-DOMツリーをデフォルト解釈バッファ103のソースおよび解釈用DOMツリー134、131にコピーし、その他、図6に示したようにして、デフォルト解釈バッファ103を初期化する（ステップS203）。

#### 【0133】

次に、このデフォルト解釈バッファ103の解釈処理をインタプリタ102で行う。ここで、例えば、図17に示したようなXML-DOMツリーを解釈するものとする。

#### 【0134】

インタプリタ102は、前述したように、命令エレメントを深さ優先で移動先のエレメントを決定していくので、図17に示すDOMツリーにおいては、まず、pz:targetsエレメントE2を解釈処理する（ステップS204～ステップS205）。その後、エレメントE1、E2の親エレメントであるpz:convertエレメントE3を解釈処理する（ステップS206～ステップS

207)。その後、図17には示していないが、`pz:convert`エレメントE3の弟エレメント、あるいは、親エレメントへ、プログラムカウンタ132を移動させて、プログラムカウンタが「NULL」になるまで、このデフォルト解釈バッファ103の解釈処理を進めていく（ステップS208）。

## 【0135】

さて、ステップS205では、`pz:targets`エレメントE2の解釈処理を行うわけだが、ここでの処理動作を図14に示す。

## 【0136】

`targets`コマンドプロセッサ122は、`pz:targets`エレメントE3の`href`属性値、すなわち、「`http://www.xxx.com/booklist.xml#xpointer(//textbook)`」を取り出し、その属性値を解釈バッファファクトリ101の入力URLとする。XMLノーマライザ111は、この入力URLにて指定された文書がXML文書でないならそれをXML文書に変換した後（ステップS212）、XML-DOMパーサ114にて、このXML文書のXML-DOMツリーを作成する（ステップS213）。なお、ここでは、当該指定された文書はXML文書であるので、そのまま、XML-DOMパーサ114にて、このXML文書のXML-DOMツリーを作成する。

## 【0137】

この場合、上記入力URLが、サーバA2のウェブ文書W2を示すXPoin ter付URLであるので、XPoin terプロセッサ115が、XPoin terフラグメント、すなわち、「`#xpointer(//textbook)`」を取り出し、ステップS213で作成されたXML-DOMツリーから当該XPoin terが指し示す「`textbook`」エレメント（その子孫エレメントを含む部分文書）のXML-DOMツリーを切り出す。「`textbook`」エレメントが複数ある場合は、それぞれに対して行う。この切り出されたXML-DOMツリーが挿入すべき部分文書のXML-DOMツリーである（ステップS214）。

## 【0138】

次に、解釈バッファイニシャライザ116により、一時解釈バッファ104を初期化し、この部分文書に `pz:targets` エlementや、`pz:convert` Elementが記述されているときは、それらの解釈処理を行って、当該部分文書のXML文書を得る。

## 【0139】

記述されていないときは、そのまま一時解釈バッファ104の解釈処理を終了し、コンテキストマネージャ121は、DOMパーサ151を用いて、当該部分文書のXML-DOMツリーからXML文書を生成し（ステップS221）、`targets` コマンドプロセッサ122は、DOMパーサ152を用いて、当該部分文書のXML文書のXML-DOMツリーを作成して、これを部分文書郡E2'として、デフォルト解釈バッファ103の解釈用XML-DOMツリー131のカレントElementである `pz:targets` Element E2と入れ替える。その結果、図18に示すように、この部分文書郡E2'が、`pz:convert` Element E3の子Elementとなり、XML-DOMツリーが更新される。生成した一時解釈バッファ104は破棄する（ステップS222）。その後、図13のステップS208へ戻る。

## 【0140】

図18に示すように、「`http://www.xxx.com/booklist.xml`」のウェブ文書内には複数の教科書データが存在するので、その全てが当該ウェブ文書の部分文書のXML-DOMツリーとして挿入されている。

## 【0141】

一方、ステップS207では、`pz:convert` Element E3の解釈処理を行うわけだが、ここでの処理動作を図15に示す。

## 【0142】

`convert` コマンドプロセッサ123は、`pz:convert` Element E3の`href` 属性値、すなわち、XSLT文書へのURL、「`textbook-book.xml`」取り出し、その属性値を解釈バッファファクトリ101の入力URLとする。以下のステップS232～ステップS240は、XLM



文書としてのXSLT文書を得るための処理であって、図14のステップS212～ステップS220と同様にして、図15のステップS241にて、図19に示したようなXML文書としてのXSLT文書を得る。

#### 【0143】

図19に示すXSLT文書は、現在の部分文書の「publication」エレメント、「price」エレメント、「author」エレメントを、それぞれ「title」エレメント、「price」エレメント、「author」エレメントへ変換するための変換ルールを記述したものである。

#### 【0144】

図19に示したようなXSLT文書を用いて、XSLTプロセッサ124は、デフォルト解釈バッファ103の解釈用XML-DOMツリー131のカレントエレメントである、pz:convertエレメントに含まれる部分文書（子部分文書とも呼ぶ）のXML-DOMツリー上の各子エレメントを変換する（ステップS242）。

#### 【0145】

ここでは、自文書内に含まれている教科書データと、「http://www.xxx.com/booklist.xml」のウェブ文書から抽出した教科書データは同じ構造のデータであるので、エレメントE1の自文書内含まれていた教科書データの場合を例にとり、図19のXSLT文書を用いて、その構造を変換する場合を説明する。

#### 【0146】

図16に示すように、エレメントE1の子エレメントである「publication」エレメントの値は、「Selected Short Stories of Shinichiro Hamada」であるが、これは、変換後では、「title」エレメントの値となる。また、図16において、エレメントE1の子エレメントである「author」エレメントの値は「Shinichiro Hamada」であるが、これは変換後では、「author」エレメントとなる。さらに、図16に示すように、エレメントE1の子エレメントである「price」エレメントの値は、「55」であるが、これは変換後も同じで

ある。

【0147】

convert コマンドプロセッサ 123 は、変換後の部分文書の XML-DOM ツリーを、新たなエレメント E3' として、デフォルト解釈バッファ 103 の解釈用 XML-DOM ツリー 131 のカレントエレメントである pz: convert エレメント E3 と入れ替えて、図 20 に示したような文書構造の XML-DOM ツリーが生成される。

【0148】

なお、生成した一時解釈バッファ 104 は破棄する（ステップ S243）。その後、図 13 のステップ S208 へ戻る。

【0149】

以上のようにして、デフォルト解釈バッファ 103 のプログラムカウンタ 132 が「NULL」となり、XML-DOM ツリー 131 の解釈が終了すると、コンテキストマネージャ 121 は、XML-DOM パーサ 151 を用いて、図 20 に示した XML-DOM ツリーを含む解釈バッファ 103 の XML-DOM ツリー 131 を基に、目的とするウェブ文書 W1 としての XML 文書を生成し出力する。

【0150】

なお、クライアント端末 B1 のウェブブラウザが XML 文書を表示できる場合は、XML 文書のウェブ文書 W1 をそのままクライアント端末 B1 のウェブブラウザに返すが、表示できない場合は、サーバ A1 側でスタイルシートを処理して、ウェブ文書 W1 を HTML 文書に変換してからクライアント端末 B1 のウェブブラウザへ返す（図 13 のステップ S209）。

【0151】

(D) ウェブ文書の合成処理のための XML-P' z サーバ間の協調動作

次に、ウェブ文書の合成処理を XML-P' z サーバ間で協調して行う場合について説明する。

【0152】

例えば、あるXML-P' zサーバ上のXML-P' z文書を解釈処理中に他のXML-P' zサーバのXML-P' z文書を挿入する場合に、その挿入されるXML-P' z文書は、どちらのサーバが解釈するのかという問題がある。すなわち、GETコマンドによる要求があった場合に、XML-P' z文書そのものを返すのか、解釈処理した結果のXML文書を返すのかという判断を行う必要があるということである。

## 【0153】

HTTPサーバ(XML-P' z文書を要求される側)とHTTPクライアント(XML-P' z文書を要求する側)との間で、HTTPクライアントがXML-P' z文書を解釈処理できない場合は、HTTPサーバ側でXML-P' z文書を解釈処理しなければならないという制約がある。

## 【0154】

この制約を判断の材料に導入するため、XML-P' z言語処理系100の解釈バッファファクトリ101が、XML-P' z文書を要求する際に、GETコマンドによる要求のヘッダに「XML-P' z: enable」をつけるものとする。

## 【0155】

また、HTTPサーバとしては、XML-P' z文書の解釈処理をHTTPクライアントに委譲することにより、サーバの負荷を下げることのできる利点もあるが、XML-P' z文書を公開したくない何らかの理由があるかもしれない(含まれている合成ロジックを公開したくないなど)ので、サーバ側でXML-P' z言語を解釈処理するかどうかは設定次第である。

## 【0156】

以上を踏まえて、HTTPサーバが解釈実行するかどうかの判断処理動作について、図10の示すフローチャートを参照して説明する。

## 【0157】

まず、ステップS71では、GET要求のヘッダに「XML-P' z: enable」が含まれているかどうかを調べ、含まれていないならば、ステップS72へ進み、HTTPサーバ上でXML-P' z文書を解釈処理して終了する。含

まれているならば、ステップ S 7 3 へ進み、HTTPサーバがXML-P' z 文書を処理する設定になっているかどうかをチェックし、そうであれば、ステップ S 7 4 へ進み、HTTPサーバでXML-P' z 文書を解釈処理して終了し、そうでなければ、ステップ S 7 5 へ進み、解釈処理をしないでHTTPクライアントにXML-P' z 文書をそのまま送信して終了する。

## 【0158】

## (E) 追記

以上説明したように、上記実施形態によれば、合成のためのベースとなる合成用ウェブ文書をXMLで記述し、指定した他のウェブ文書から指定した範囲の部分（部分文書）を抽出して、それを合成用ウェブ文書の指定された位置に挿入し、合成用ウェブ文書の指定した範囲に変換処理を施す、挿入・変換の2つの合成ロジック命令をその合成用ウェブ文書内にエレメントとして持たせたXML-P' z (XML-P i e c e s) 文書を定義する。言語処理系100は、XML-P' z 文書に記述されている、指定されたウェブサーバ（例えば、ここでは、ウェブサーバA2、A3）のウェブ文書（ページ）W2、W3から指定した範囲の部分（部分文書）を抽出し、それをXML-P' z 文書の指定位置に挿入するとともに、XML-P' z 文書に記述されている指定された範囲に変換処理を施す。最終的に、XML-P' z 言語処理系100の処理結果としてのXML文書（合成されたウェブ文書）W1を得ることにより、複数のウェブサイトの情報を1つのウェブ文書上に合成することが容易にしかも汎用的に行える。

## 【0159】

なお、上記実施形態に記載した手法は、コンピュータに実行させることのできるプログラムとして、DVD、CD-ROM、フロッピディスク、個体メモリ、光ディスクなどの記録媒体に格納して頒布することもできる。

## 【0160】

## 【発明の効果】

以上説明したように、本発明によれば、複数のウェブサイトの情報を1つのウェブ文書上に合成することが容易にしかも汎用的に行える。

【図面の簡単な説明】

【図 1】

本発明のXML-P' z 言語処理系を組み込んだウェブサーバ (XML-P' z サーバ) の基本的な動作を説明するための図。

【図 2】

XML-P' z 言語処理系の全体の構成例を示した図。

【図 3】

HTML判定器において、与えられたURLにて指定されるウェブ文書がHTML文書かXML文書かを判定するための処理動作を示したフローチャート。

【図 4】

HTML-XMLコンバータのHTML文書からXML文書への変換処理動作を説明するためのフローチャート。

【図 5】

XPointerプロセッサのXPointerフラグメントに対する処理動作を説明するためのフローチャート。

【図 6】

解釈バッファイニシャライザの解釈バッファの初期化処理動作を説明するためのフローチャート。

【図 7】

コンテキストマネージャの処理動作を説明するためのフローチャート。

【図 8】

targetsコマンドプロセッサのtargets要素の解釈処理動作を説明するためのフローチャート。

【図 9】

convertコマンドプロセッサのconvert要素の解釈処理動作を説明するためのフローチャート。

【図 10】

XML-P' z 文書の解釈処理をサーバ側で行うかクライアント側で行うかを判断する判断処理動作について説明するためのフローチャート。

【図 1 1】

(a) 図は、XML-P' z 文書の第 1 の例の文書構造を模式的に示した図で、  
(b) 図は、XML-P' z 文書の解釈後の XML 文書の文書構造を示した図

【図 1 2】

XML-P' z 文書の解釈順序について説明するための図。

【図 1 3】

図 2 に示した構成の言語処理系が、複数のウェブ文書を 1 つのウェブ文書上に合成するための連の動作を説明するためのフローチャート。

【図 1 4】

図 2 に示した構成の言語処理系が、複数のウェブ文書を 1 つのウェブ文書上に合成するための連の動作を説明するためのフローチャート。

【図 1 5】

図 2 に示した構成の言語処理系が、複数のウェブ文書を 1 つのウェブ文書上に合成するための連の動作を説明するためのフローチャート。

【図 1 6】

合成用ウェブ文書としての XML-P' z 文書の一例であって、XML-P' z 文書の一部を示した図。

【図 1 7】

図 1 6 の XML-P' z 文書に対応する XML-DOM ツリーを概略的に示した図。

【図 1 8】

図 1 6 の pz:targets エレメントを解釈した結果の XML-DOM ツリーを概略的に示した図。

【図 1 9】

図 1 6 の XML-P' z 文書に記述されている XSLT 文書の一例を示した図

【図 2 0】

図 1 6 の pz:targets エレメントと pz:convert エレメント

を解釈した結果のXML-DOMツリーを概略的に示した図。

【符号の説明】

- A 1、A 2、A 3…サーバ
- B 1…クライアント端末
- W 1…合成されたウェブ文書（XML文書）
- W 2～W 3…ウェブ文書
- 1…XML-P' z 言語処理系（合成処理部）
- 2…XML-P' z 文書
- 1 0 0…XML-P' z 言語処理系
- 1 0 1…解釈バッファファクトリ
- 1 0 2…インタプリタ
- 1 0 3…デフォルト解釈バッファ
- 1 0 4…一時解釈バッファ
- 1 1 1…XMLノーマライザ
- 1 1 2…HTML判定器
- 1 1 3…HTML-XMLコンバータ
- 1 1 4…XML-DOMパーサ
- 1 1 5…X P o i n t e r プロセッサ
- 1 1 6…解釈バッファイニシャライザ
- 1 2 1…コンテキストマネージャ
- 1 2 2…t a r g e t s コマンドマネージャ
- 1 2 3…c o n v e r t コマンドマネージャ
- 1 2 4…X S L T プロセッサ
- 1 3 1…解釈用XML-DOMツリー
- 1 3 2…プログラムカウンタ
- 1 3 3…ロードフラグ
- 1 3 4…ソースXML-DOMツリー
- 1 4 1…解釈用XML-DOMツリー
- 1 4 2…プログラムカウンタ

1 4 3 …ロードフラグ

1 4 4 …ソースXML-DOMツリー

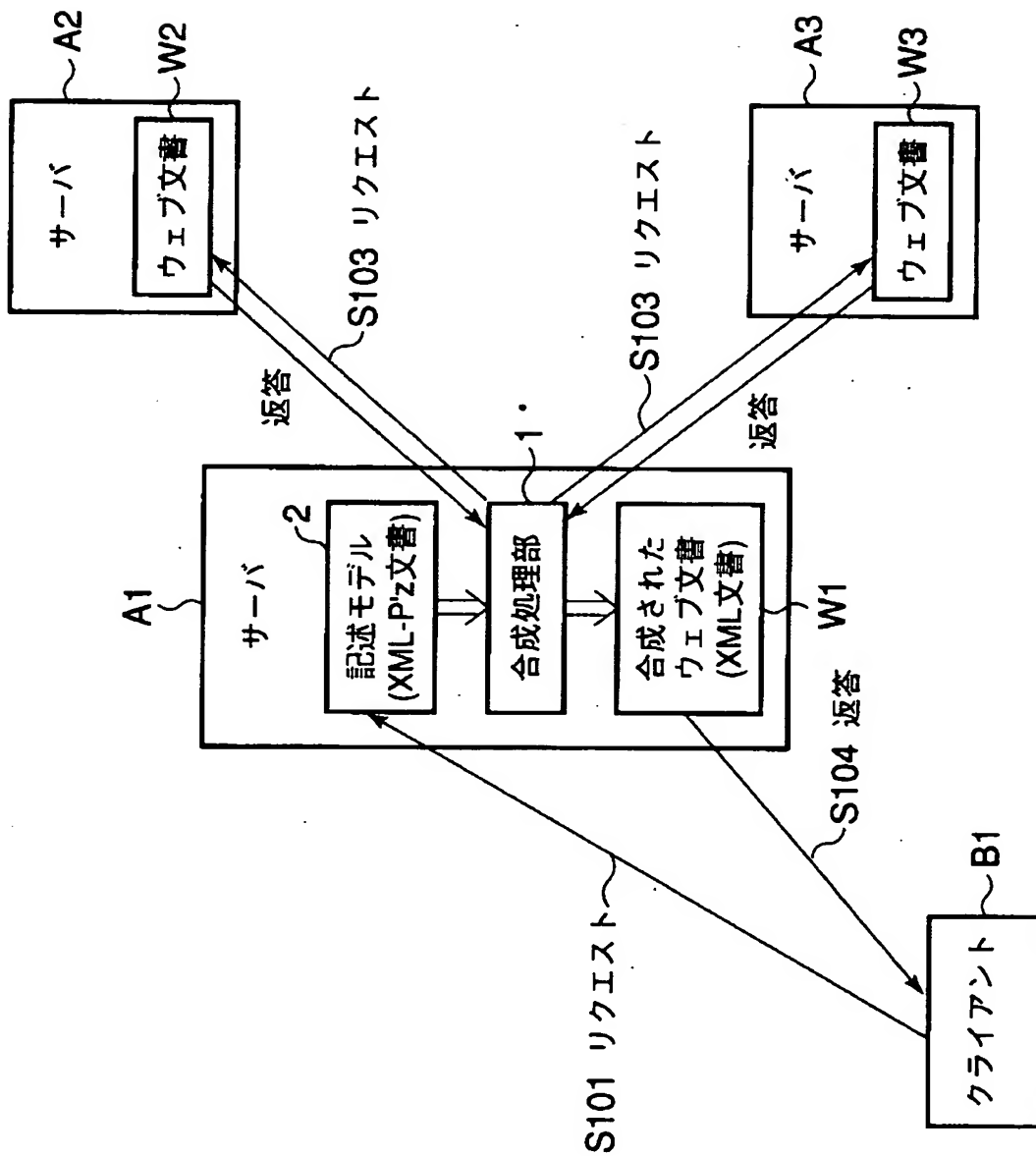
1 5 1 ~ 1 5 3 …DOMパーサ



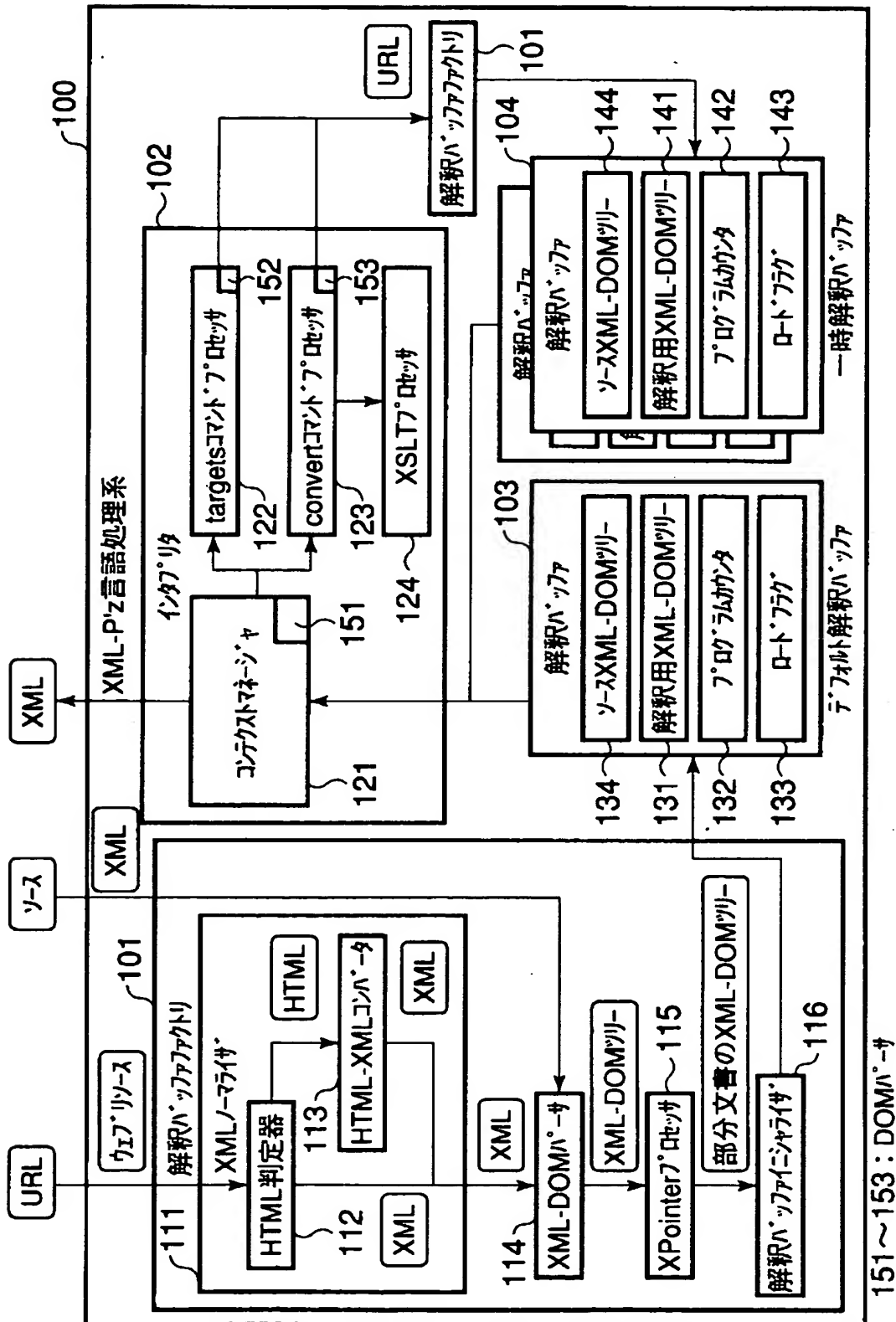
【書類名】

図面

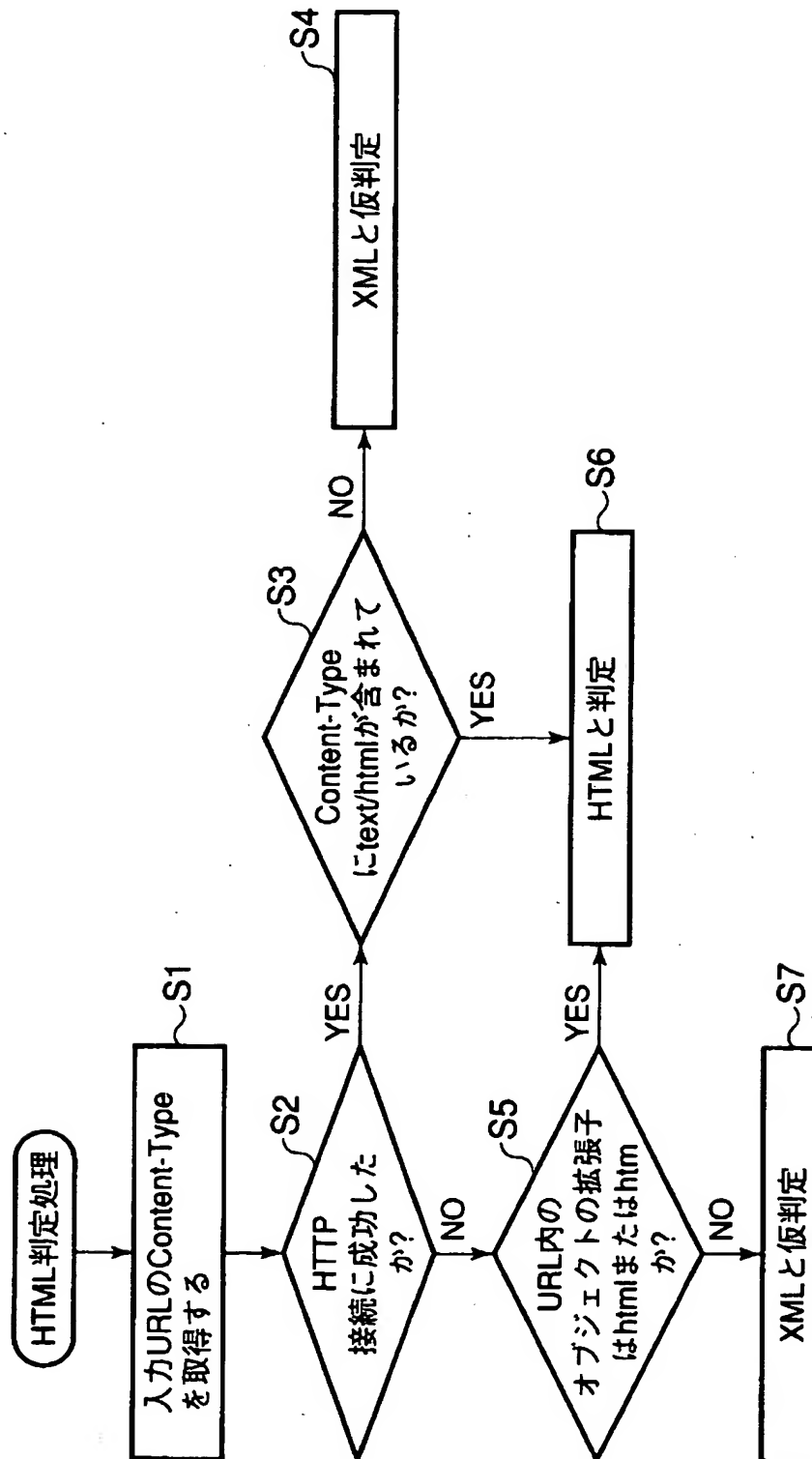
【図 1】



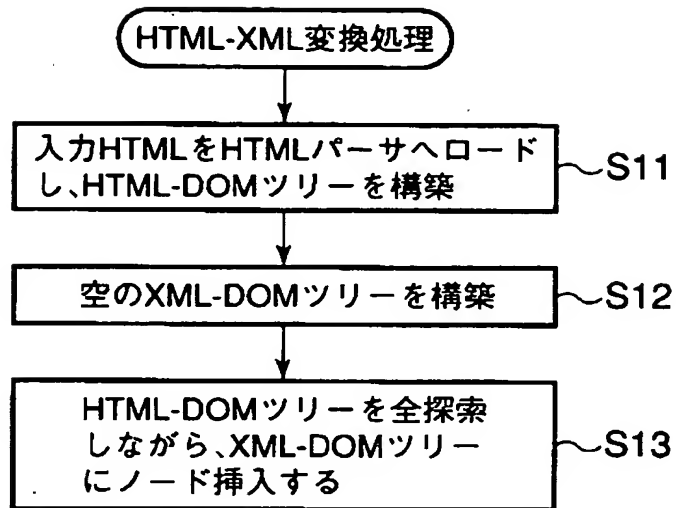
【図2】



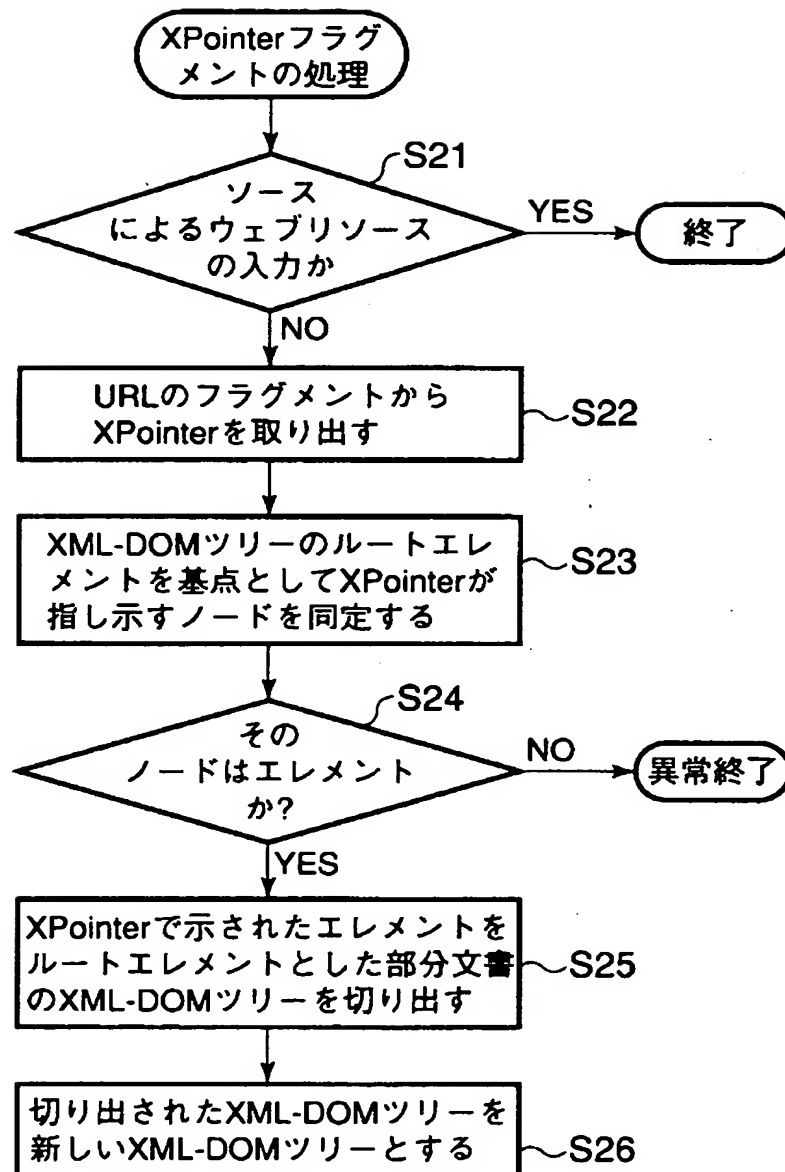
【図3】



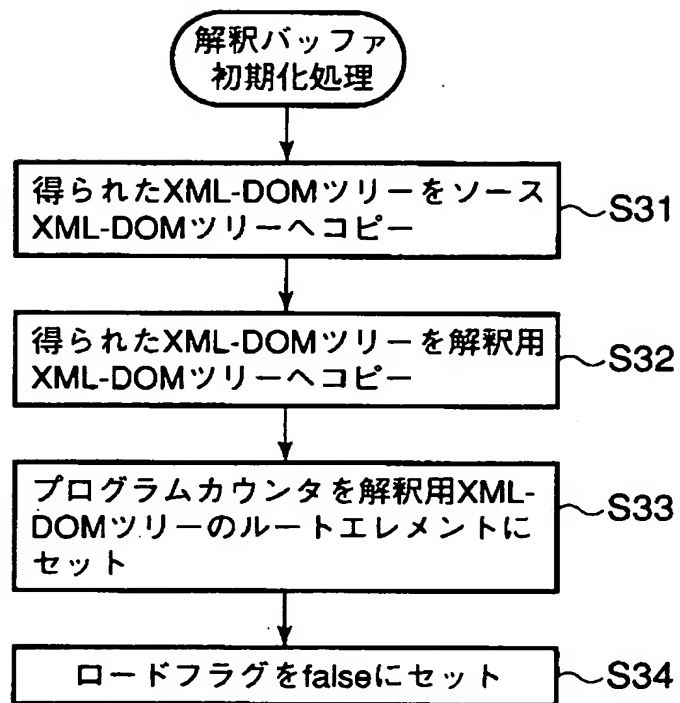
【図 4】



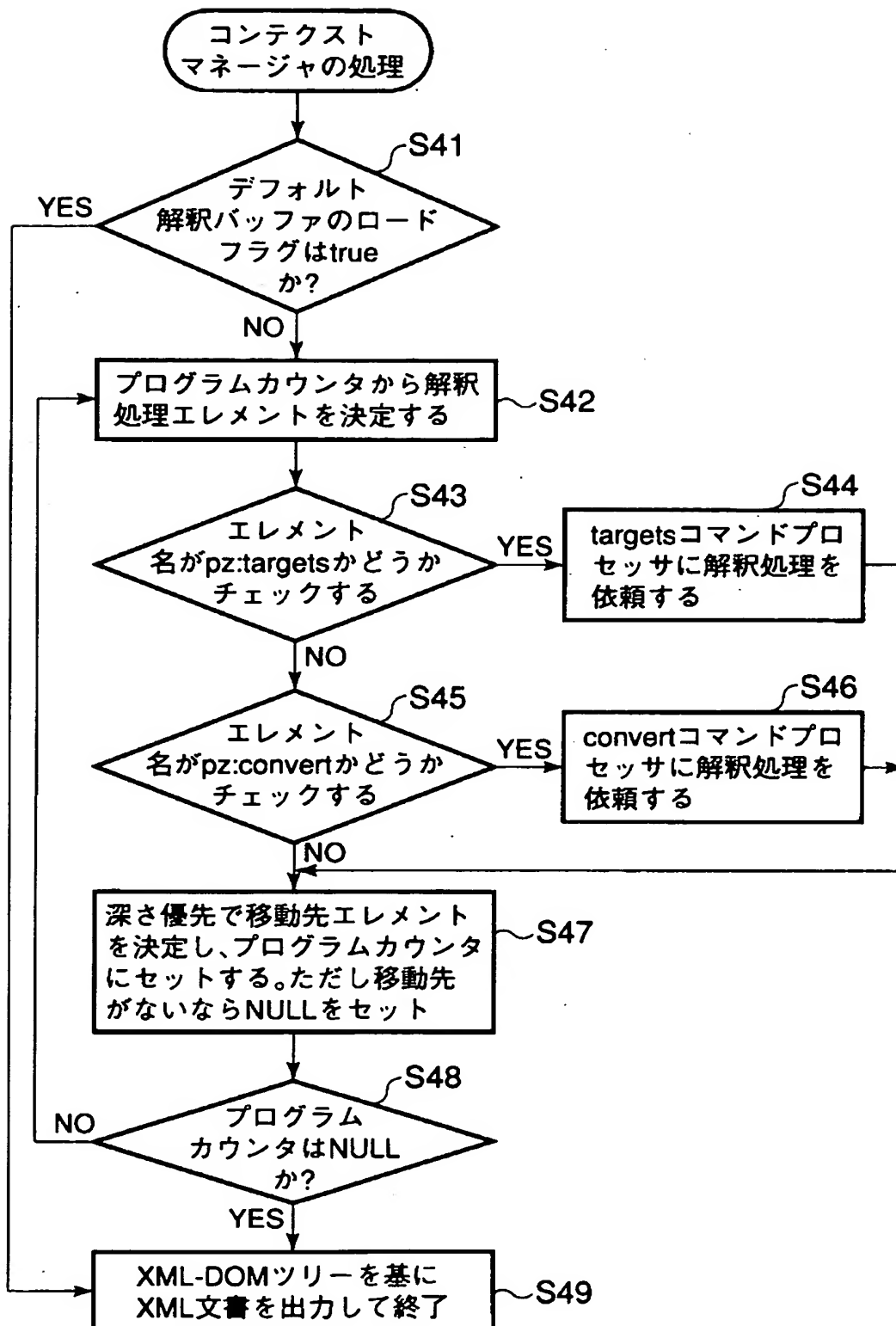
【図 5】



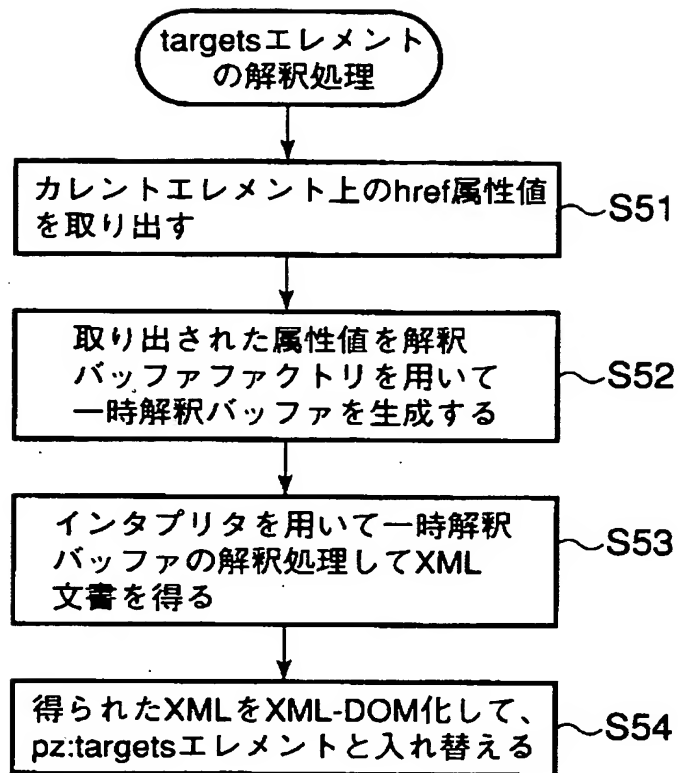
【図 6】



【図 7】

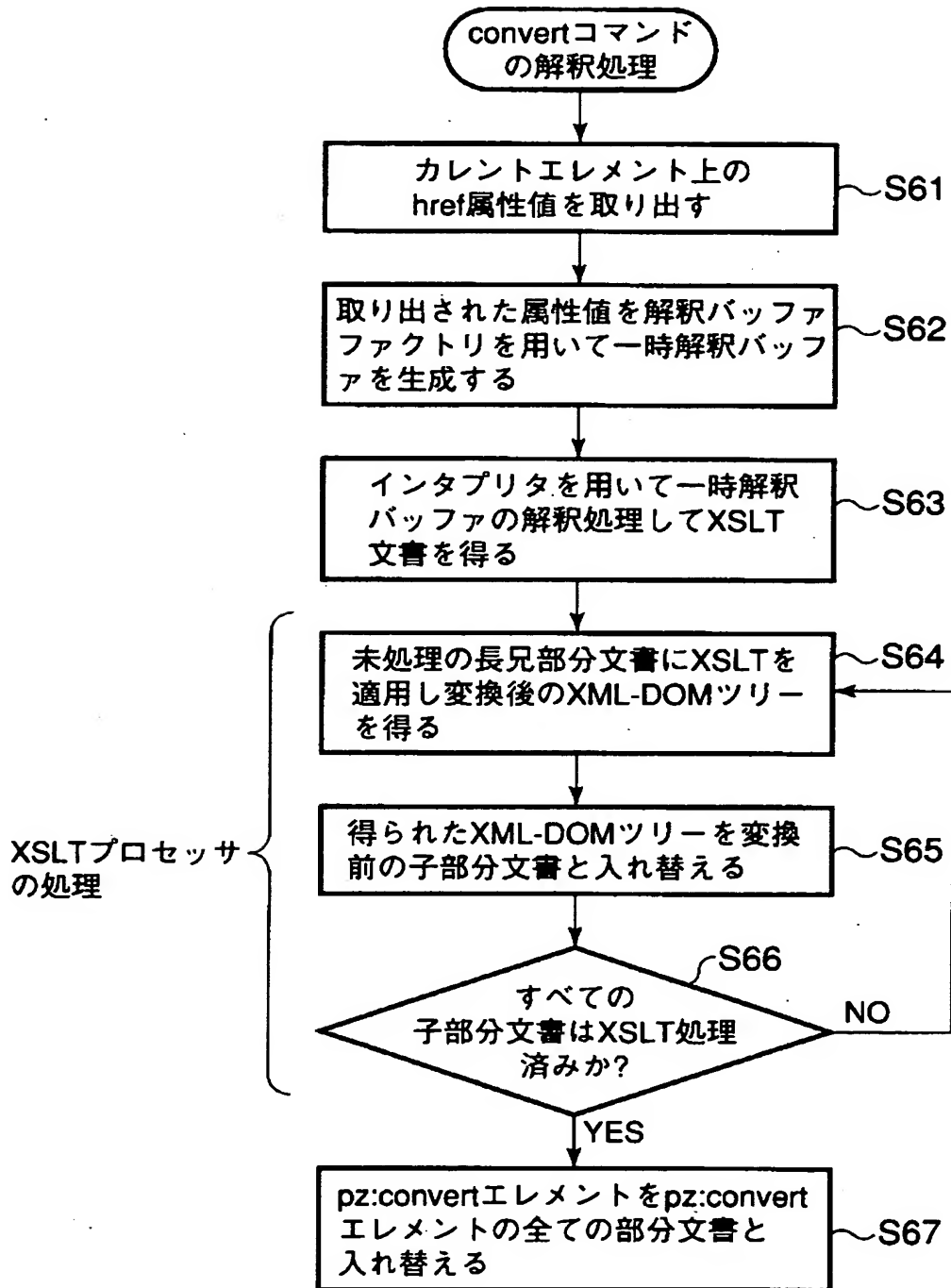


【図 8】

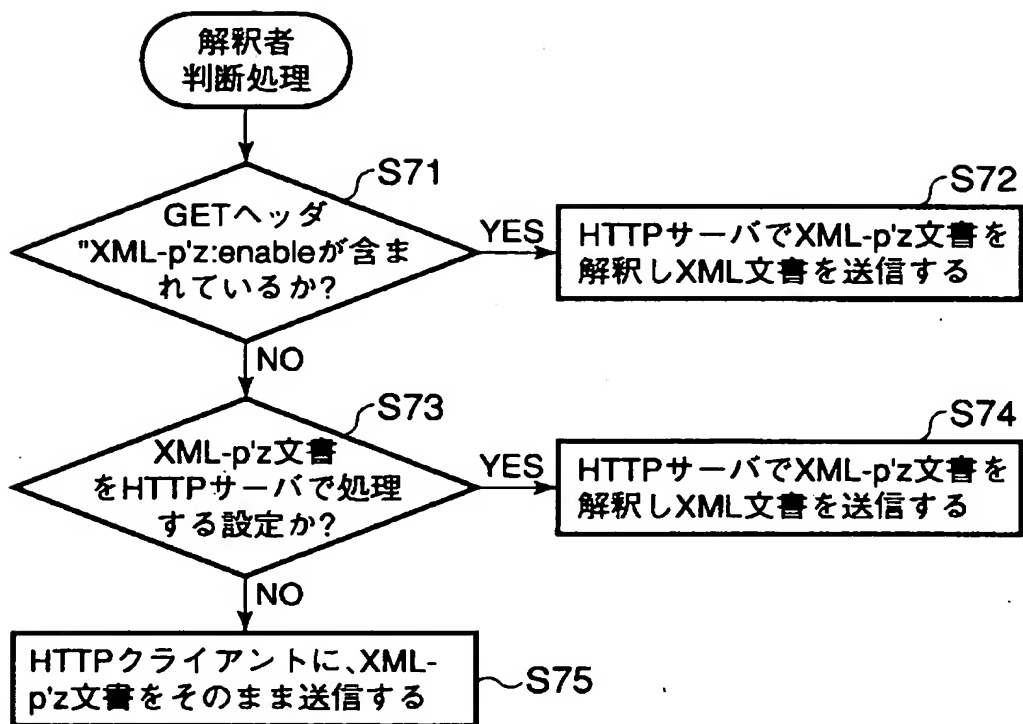




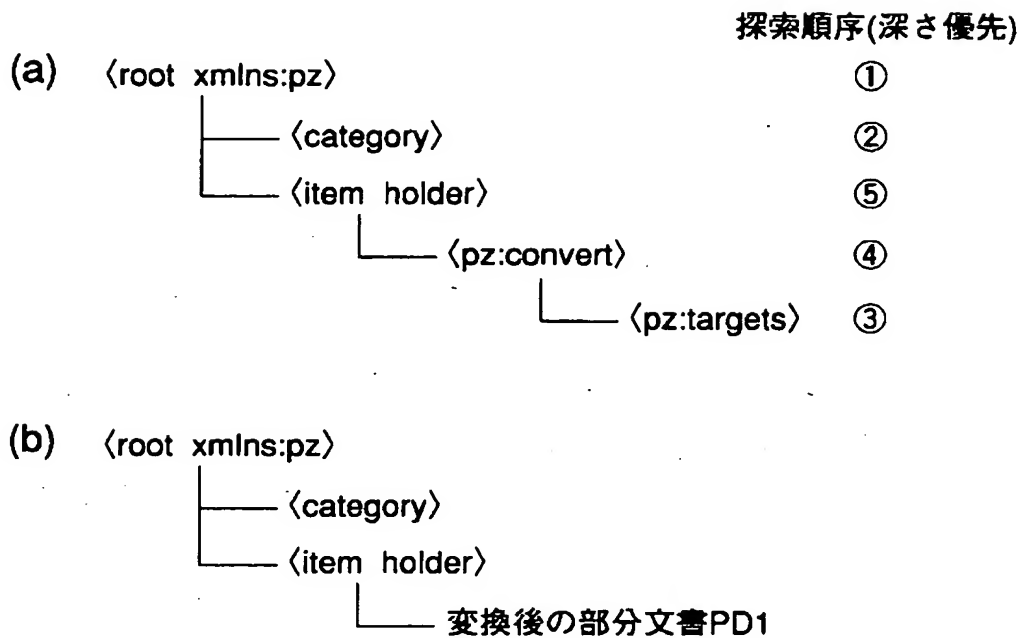
【図 9】



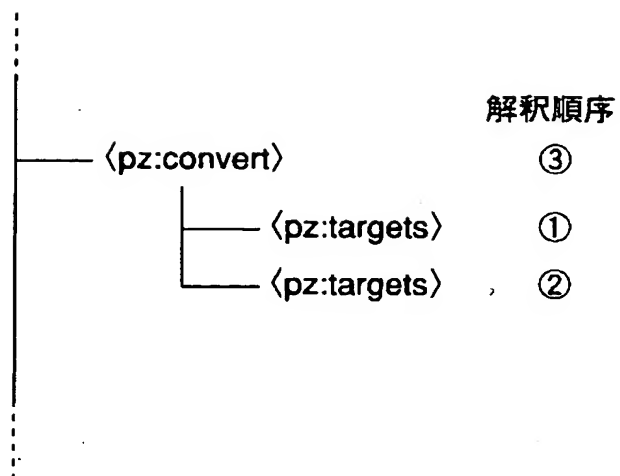
【図 1 0】



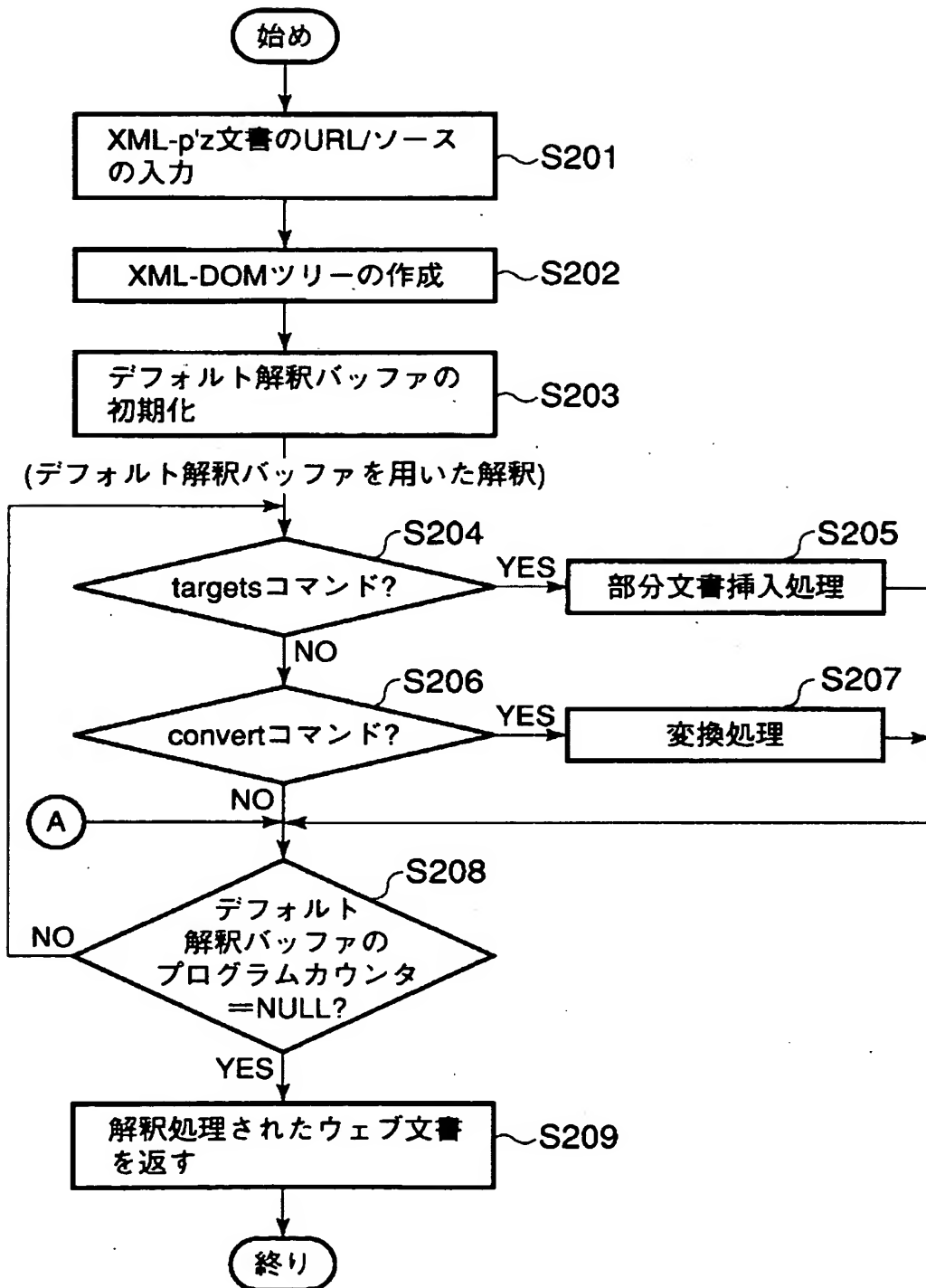
【図 1 1】



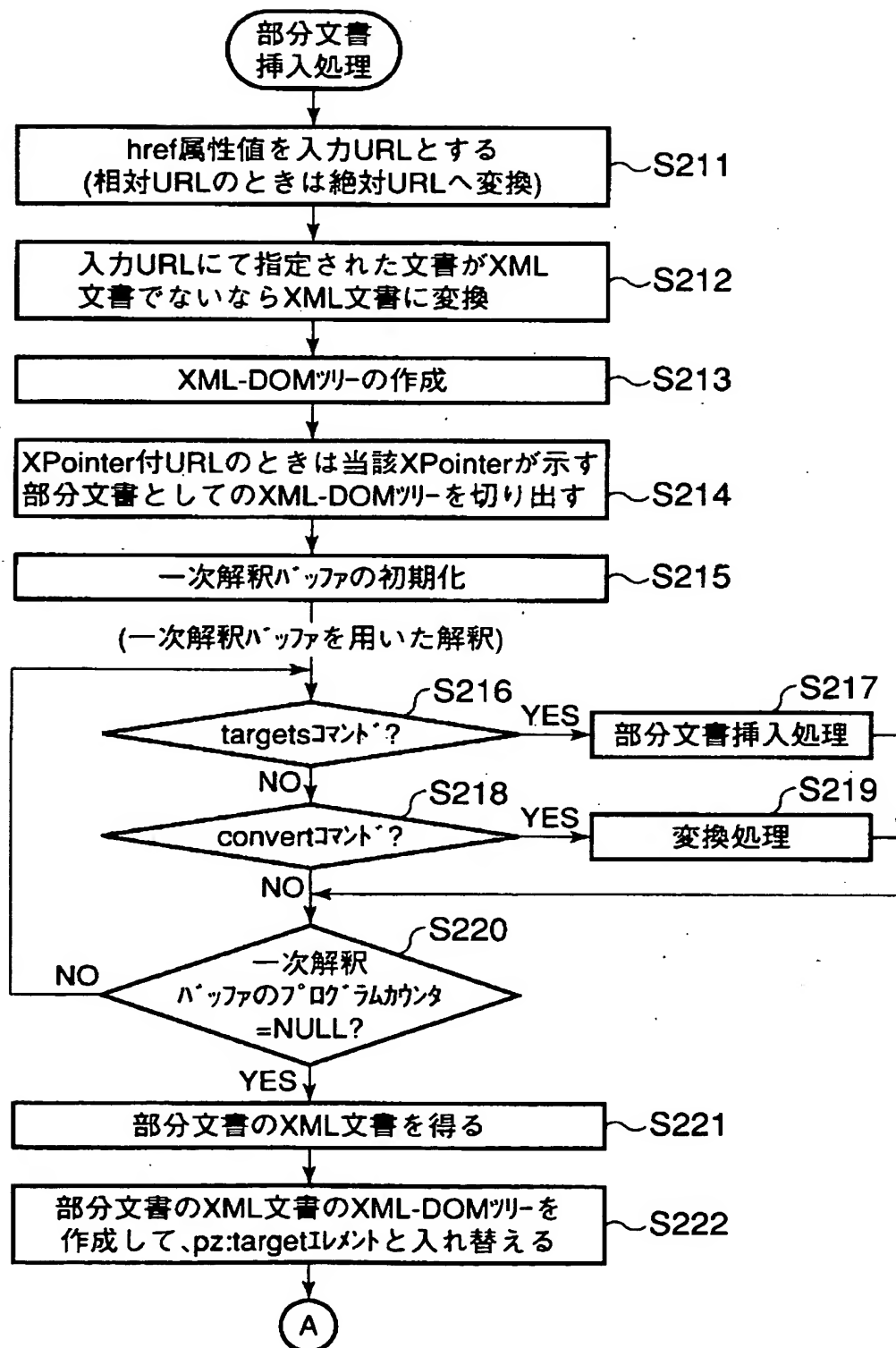
【図 1 2】



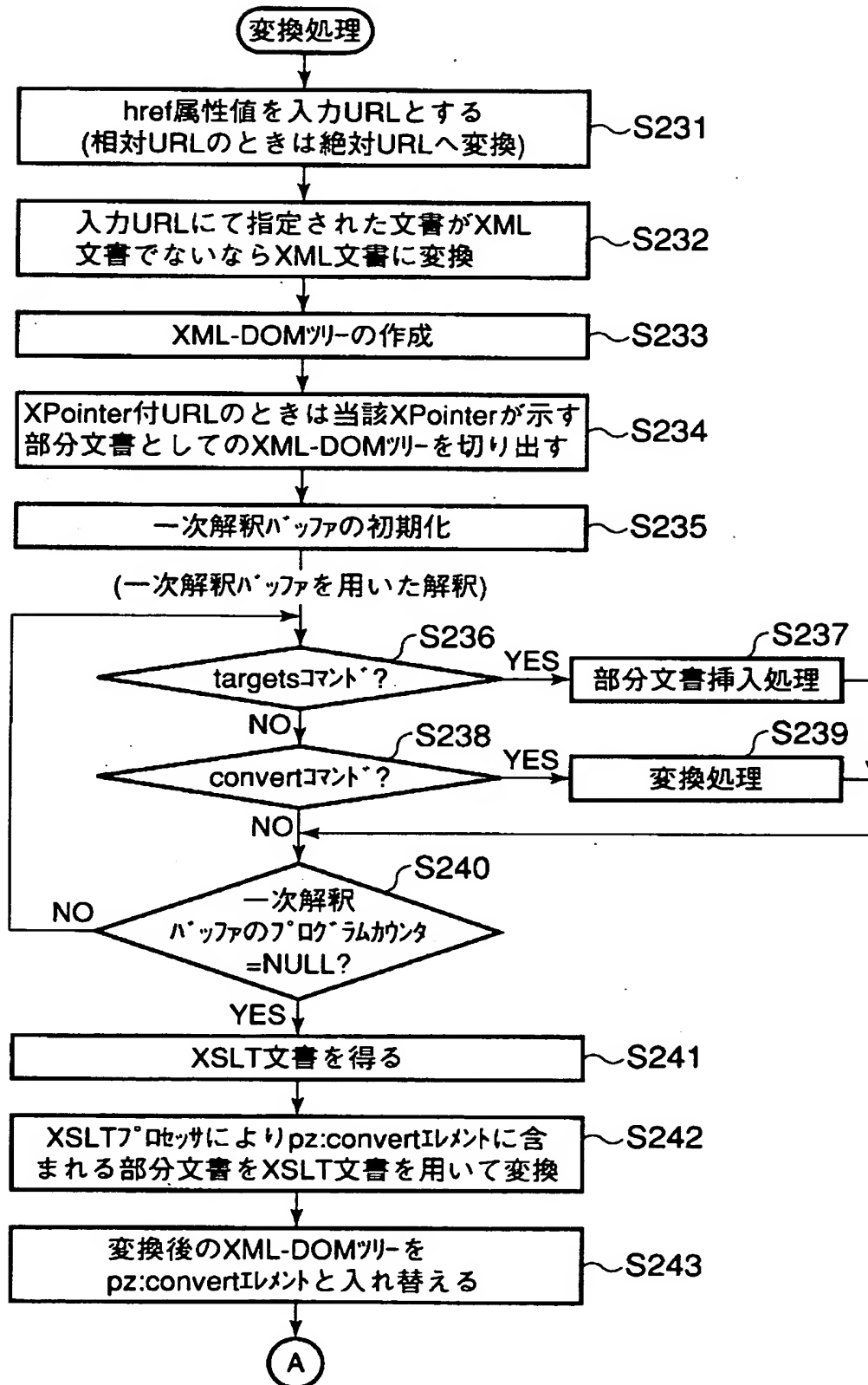
【図13】



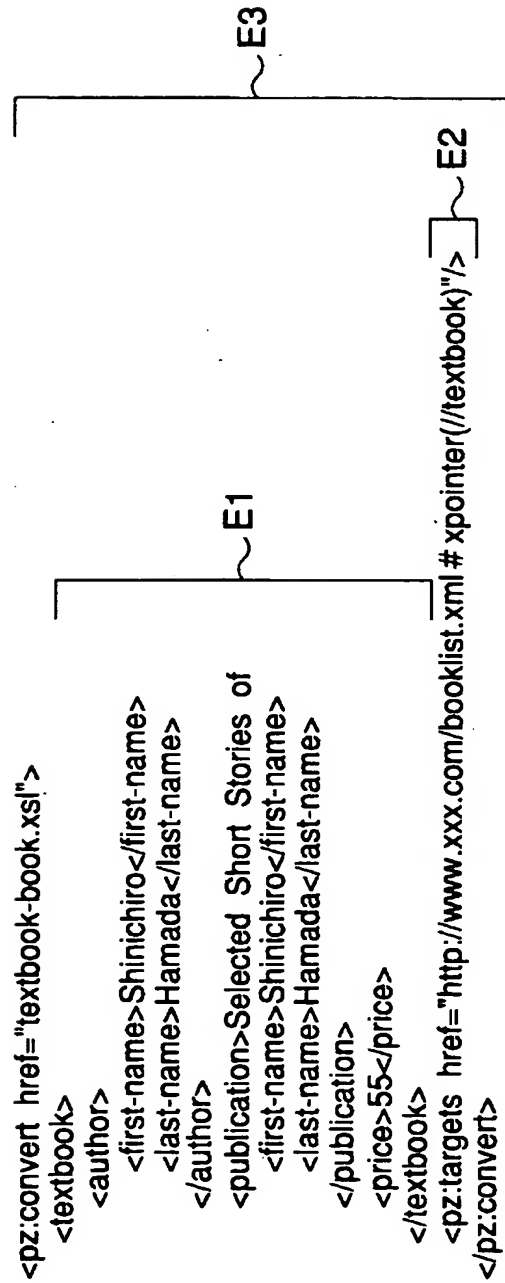
【図 1 4】



【図 1 5】

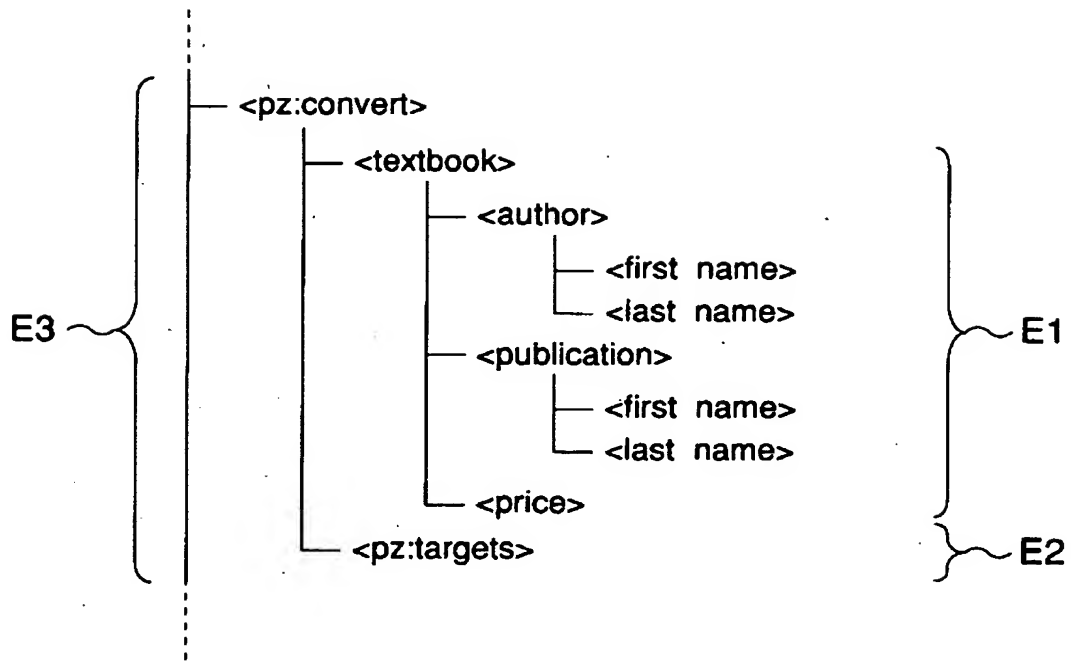


【図 1 6】

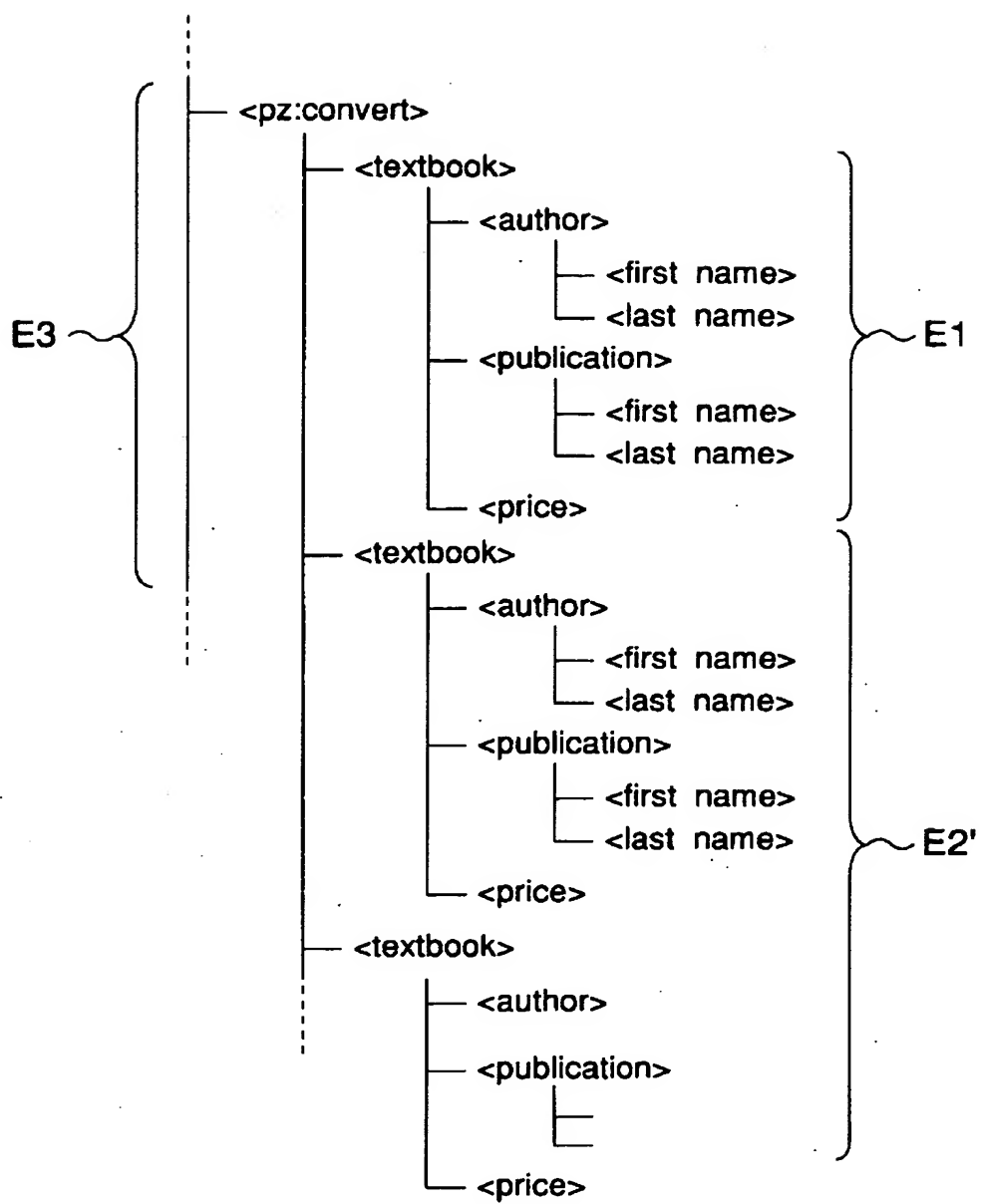




【図 1 7】



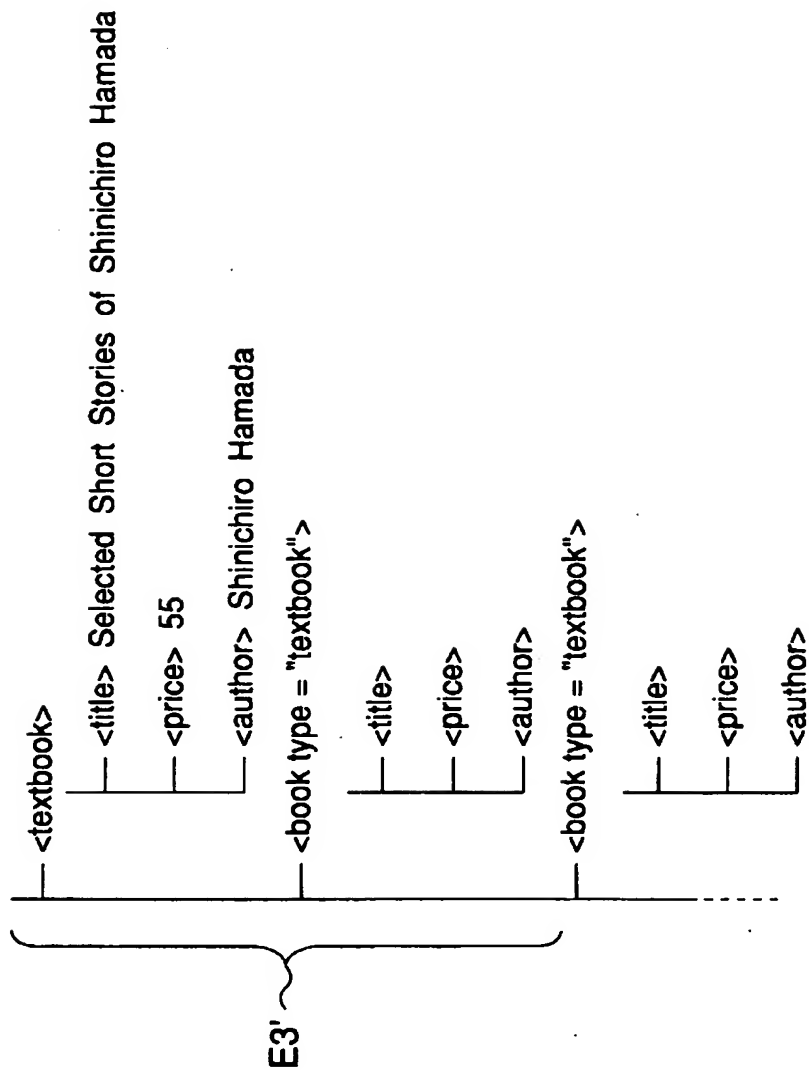
【図 1 8】



【図 1 9】

```
textbook-book.xsl
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" encoding="Shift_JIS" />
  <xsl:template match="text()" />
  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="/textbook">
    <book type="textbook">
      <title> <xsl:value-of select="publication"/> </title>
      <price> <xsl:value-of select="price"/> </price>
      <author> <xsl:value-of select="author"/> </author>
    </book>
  </xsl:template>
</xsl:stylesheet>
```

【図 2 0】



【書類名】 要約書

【要約】

【課題】複数のウェブサイトの情報を1つのウェブ文書上に合成することが容易にしかも汎用的に行える文書合成方法および文書合成装置を提供する。

【解決手段】少なくとも、インターネットにおけるWWW上のマークアップ言語で記述された第1の文書のインターネット上の所在と、第1の文書から抽出する部分文書の範囲と、合成用の第2の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第2の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第2の文書に従って、前記第1の文書から前記部分文書を抽出して、その部分文書を前記第2の文書上の前記指定された合成位置に挿入するとともに、前記変換ルールを用いて前記第2の文書上の前記指定された範囲の文書構造を変換する。

【選択図】 図2

出 願 人 履 歴 情 報

識別番号 [000003078]

- |          |                  |
|----------|------------------|
| 1. 変更年月日 | 1990年 8月22日      |
| [変更理由]   | 新規登録             |
| 住 所      | 神奈川県川崎市幸区堀川町72番地 |
| 氏 名      | 株式会社東芝           |
| 2. 変更年月日 | 2001年 7月 2日      |
| [変更理由]   | 住所変更             |
| 住 所      | 東京都港区芝浦一丁目1番1号   |
| 氏 名      | 株式会社東芝           |